

Tal Malkin
Chris Peikert (Eds.)

LNCS 12828

Advances in Cryptology – CRYPTO 2021

41st Annual International Cryptology Conference, CRYPTO 2021
Virtual Event, August 16–20, 2021
Proceedings, Part IV

4
Part IV



 Springer

Founding Editors

Gerhard Goos

Karlsruhe Institute of Technology, Karlsruhe, Germany

Juris Hartmanis

Cornell University, Ithaca, NY, USA

Editorial Board Members

Elisa Bertino

Purdue University, West Lafayette, IN, USA

Wen Gao

Peking University, Beijing, China

Bernhard Steffen 

TU Dortmund University, Dortmund, Germany

Gerhard Woeginger 

RWTH Aachen, Aachen, Germany

Moti Yung

Columbia University, New York, NY, USA

More information about this subseries at <http://www.springer.com/series/7410>


Tal Malkin · Chris Peikert (Eds.)

Advances in Cryptology – CRYPTO 2021

41st Annual International Cryptology Conference, CRYPTO 2021
Virtual Event, August 16–20, 2021
Proceedings, Part IV

Editors

Tal Malkin 
Columbia University
New York City, NY, USA

Chris Peikert 
University of Michigan
Ann Arbor, MI, USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-84258-1 ISBN 978-3-030-84259-8 (eBook)
<https://doi.org/10.1007/978-3-030-84259-8>

LNCS Sublibrary: SL4 – Security and Cryptology

© International Association for Cryptologic Research 2021

This work is subject to copyright. All rights are reserved 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 Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 41st International Cryptology Conference (Crypto 2021), sponsored by the International Association of Cryptologic Research (IACR), was held during August 16–20, 2021. Due to the ongoing COVID-19 pandemic, and for the second consecutive year, Crypto was held as an online-only virtual conference, instead of at its usual venue of the University of California, Santa Barbara. In addition, six affiliated workshop events took place during the days immediately prior to the conference.

The Crypto conference continues its substantial growth pattern: this year’s offering received a record-high 430 submissions for consideration, of which 103 (also a record) were accepted to appear in the program. The two program chairs were not allowed to submit a paper, and Program Committee (PC) members were limited to two submissions each. Review and extensive discussion occurred from late February through mid-May, in a double-blind, two-stage process that included an author rebuttal phase (following the initial reviews) and extensive discussion by reviewers. We thank the 58-person PC and the 390 external reviewers for their efforts to ensure that, during the continuing COVID-19 pandemic and unusual work and life circumstances, we nevertheless were able to perform a high-quality review process.

The PC selected four papers to receive recognition via awards, along with invitations to the Journal of Cryptology, via a voting-based process that took into account conflicts of interest (the program chairs did not vote).

- The Best Paper Award went to “On the Possibility of Basing Cryptography on $EXP \neq BPP$ ” by Yanyi Liu and Rafael Pass.
- The Best Paper by Early Career Researchers Award, along with an Honorable Mention for Best Paper, went to “Linear Cryptanalysis of FF3-1 and FEA” by Tim Beyne.
- Honorable Mentions for Best Paper also went to “Efficient Key Recovery for all HFE Signature Variants” by Chengdong Tao, Albrecht Petzoldt, and Jintai Ding; and “Three Halves Make a Whole? Beating the Half-Gates Lower Bound for Garbled Circuits” by Mike Rosulek and Lawrence Roy.

In addition to the regular program, Crypto 2021 included two invited talks, by Vanessa Teague on “Which e-voting problems do we need to solve?” and Jens Groth on “A world of SNARKs.” The conference also carried forward the long-standing tradition of having a rump session, organized in a virtual format.

The chairs would also like to thank the many other people whose hard work helped ensure that Crypto 2021 was a success:

- Vladimir Kolesnikov (Georgia Institute of Technology)—Crypto 2021 general chair.
- Daniele Micciancio (University of California, San Diego), Thomas Ristenpart (Cornell Tech), Yevgeniy Dodis (New York University), and Thomas Shrimpton (University of Florida)—Crypto 2021 Advisory Committee.

- Carmit Hazay (Bar Ilan University)—Crypto 2021 workshop chair.
- Bertram Poettering and Antigoni Polychroniadou—Crypto 2021 rump session chairs.
- Kevin McCurley, for his critical assistance in setting up and managing the HotCRP paper submission and review system, conference website, and other technology.
- Kevin McCurley, Kay McKelly, and members of the IACR’s emergency pandemic team for their work in designing and running the virtual format.
- Anna Kramer and her colleagues at Springer.

July 2021

Tal Malkin
Chris Peikert

Organization

General Chair

Vladimir Kolesnikov Georgia Institute of Technology, USA

Program Committee Chairs

Tal Malkin Columbia University, USA
Chris Peikert University of Michigan and Algorand, Inc., USA

Program Committee

Abhi Shelat Northeastern University, USA
Andrej Bogdanov Chinese University of Hong Kong, Hong Kong
Antigoni Polychroniadou JP Morgan AI Research, USA
Brice Minaud Inria and École Normale Supérieure, France
Chaya Ganesh Indian Institute of Science, India
Chris Peikert University of Michigan and Algorand, Inc., USA
Claudio Orlandi Aarhus University, Denmark
Daniele Venturi Sapienza University of Rome, Italy
David Cash University of Chicago, USA
David Wu University of Virginia, USA
Dennis Hofheinz ETH Zurich, Switzerland
Divesh Aggarwal National University of Singapore, Singapore
Dominique Unruh University of Tartu, Estonia
Elena Andreeva Technical University of Vienna, Austria
Elena Kirshanova Immanuel Kant Baltic Federal University, Russia
Fabrice Benhamouda Algorand Foundation, USA
Fang Song Portland State University, USA
Frederik Vercauteren KU Leuven, Belgium
Ghada Almashaqbeh University of Connecticut, USA
Itai Dinur Ben-Gurion University, Israel
Jean-Pierre Tillich Inria, France
Jeremiah Blocki Purdue University, USA
John Schanck University of Waterloo, Canada
Jonathan Bootle IBM Research, Switzerland
Joseph Jaeger University of Washington, USA
Junqing Gong East China Normal University, China
Lisa Kohl CWI Amsterdam, The Netherlands
Manoj Prabhakaran IIT Bombay, India
Marcel Keller CSIRO's Data61, Australia
Mariana Raykova Google, USA

Mike Rosulek	Oregon State University, USA
Mor Weiss	Bar-Ilan University, Israel
Muthuramakrishnan Venkitasubramaniam	University of Rochester, USA
Ni Trieu	Arizona State University, USA
Nir Bitansky	Tel Aviv University, Israel
Nuttapong Attrapadung	AIST, Japan
Omer Paneth	Tel Aviv University, Israel
Paul Grubbs	NYU, Cornell Tech and University of Michigan, USA
Peihan Miao	University of Illinois at Chicago, USA
Peter Schwabe	Max Planck Institute for Security and Privacy, Germany, and Radboud University, The Netherlands
Ran Canetti	BU, USA, and Tel Aviv University, Israel
Romain Gay	IBM Research, Switzerland
Ron Steinfeld	Monash University, Australia
Rosario Gennaro	City University of New York, USA
Ryo Nishimaki	NTT Secure Platform Laboratories, Japan
Sandro Coretti	IOHK, Switzerland
Sikhar Patranabis	Visa Research, USA
Sina Shiehian	UC Berkeley and Stony Brook University, USA
Siyao Guo	NYU Shanghai, China
Stanislaw Jarecki	University of California, Irvine, USA
Tal Malkin	Columbia University, USA
Tarik Moataz	Aroki Systems, USA
Thomas Peters	UC Louvain, Belgium
Thomas Peyrin	Nanyang Technological University, Singapore
Tianren Liu	University of Washington, USA
Viet Tung Hoang	Florida State University, USA
Xavier Bonnetain	University of Waterloo, Canada
Yu Yu	Shanghai Jiao Tong University, China

Additional Reviewers

Aaram Yun	Akshayaram Srinivasan
Aarushi Goel	Akshima
Aayush Jain	Alain Passelègue
Abhishek Jain	Alex Bienstock
Adrien Benamira	Alex Lombardi
Agnes Kiss	Alexander Golovnev
Aishwarya Thiruvengadam	Alexander Hoover
Ajith Suresh	Alexander May
Akin Ünal	Alexandre Wallet
Akinori Kawachi	Alexandru Cojocaru
Akira Takahashi	Alice Pellet-Mary
Akshay Degwekar	Alin Tomescu

Amin Sakzad
Amit Singh Bhati
Amitabh Trehan
Amos Beimel
Anat Paskin-Cherniavsky
Anca Nitulescu
André Chailloux
Andre Esser
André Schrottenloher
Andrea Coladangelo
Andreas Hülsing
Antonin Leroux
Antonio Florez-Gutierrez
Archita Agarwal
Ariel Hamlin
Arka Rai Choudhuri
Arnab Roy
Ashrujit Ghoshal
Ashutosh Kumar
Ashwin Jha
Atsushi Takayasu
Aurore Guillevic
Avijit Dutta
Avishay Yanay
Baiyu Li
Balazs Udvarhelyi
Balthazar Bauer
Bart Mennink
Ben Smith
Benjamin Diamond
Benjamin Fuller
Benny Applebaum
Benoît Cogliati
Benoit Libert
Bertram Poettering
Binyi Chen
Bo-Yin Yang
Bogdan Ursu
Bruno Freitas dos Santos
Bryan Parno
Byeonghak Lee
Carl Bootland
Carles Padro
Carmit Hazay
Carsten Baum
Cecilia Boschini
Chan Nam Ngo
Charles Momin
Charlotte Bonte
Chen Qian
Chen-Da Liu-Zhang
Chenkai Weng
Chethan Kamath
Chris Brzuska
Christian Badertscher
Christian Janson
Christian Majenz
Christian Matt
Christina Boura
Christof Paar
Christoph Egger
Cody Freitag
Dahmun Goudarzi
Dakshita Khurana
Damian Vizar
Damiano Abram
Damien Stehlé
Damien Vergnaud
Daniel Escudero
Daniel Jost
Daniel Masny
Daniel Tschudi
Daniel Wichs
Dario Catalano
Dario Fiore
David Gerault
David Heath
Debbie Leung
Dean Doron
Debapriya Basu Roy
Dima Kogan
Dimitrios Papadopoulos
Divya Gupta
Divya Ravi
Dominique Schröder
Eduardo Soria-Vazquez
Eldon Chung
Emmanuela Orsini
Eran Lambooj
Eran Omri
Eshan Chattopadhyay
Estuardo Alpirez Bock

Evgenios Kornaropoulos
 Eysa Lee
 Fabio Banfi
 Felix Engelmann
 Felix Günther
 Ferdinand Sibleyras
 Fermi Ma
 Fernando Virdia
 Francesco Berti
 François-Xavier Standaert
 Fuyuki Kitagawa
 Gaëtan Cassiers
 Gaëtan Leurent
 Gayathri Annapurna Garimella
 Geoffroy Couteau
 Georg Fuchsbauer
 Ghouas Amjad
 Gildas Avoine
 Giorgos Panagiotakos
 Giorgos Zirdelis
 Giulio Malavolta
 Guy Rothblum
 Hamidreza Khoshakhlagh
 Hamza Abusalah
 Hanjun Li
 Hannah Davis
 Haoyang Wang
 Hart Montgomery
 Henry Corrigan-Gibbs
 Hila Dahari
 Huijia Lin
 Ian McQuoid
 Ignacio Cascudo
 Igors Stepanovs
 Ilan Komargodski
 Ilia Iliashenko
 Ingrid Verbauwhede
 Itamar Levi
 Ittai Abraham
 Ivan Damgård
 Jack Doerner
 Jacob Schuldt
 James Bartusek
 Jan Czajkowski
 Jan-Pieter D’Anvers
 Jaspal Singh

Jean Paul Degabriele
 Jesper Buus Nielsen
 Jesús-Javier Chi-Domínguez
 Ji Luo
 Jian Guo
 Jiaxin Pan
 Jiayu Xu
 Joanne Adams-Woodage
 João Ribeiro
 Joël Alwen
 Julia Hesse
 Julia Len
 Julian Loss
 Junichi Tomida
 Justin Holmgren
 Justin Thaler
 Kai-Min Chung
 Katerina Sotiraki
 Katharina Boudgoust
 Kathrin Hövelmanns
 Katsuyuki Takashima
 Kazuhiko Minematsu
 Keita Xagawa
 Kevin Yeo
 Kewen Wu
 Khoa Nguyen
 Koji Nuida
 Kristina Hostáková
 Laasya Bangalore
 Lars Knudsen
 Lawrence Roy
 Lejla Batina
 Lennart Braun
 Léo Colisson
 Leo de Castro
 Léo Ducas
 Léo Perrin
 Lin Lyu
 Ling Song
 Luca De Feo
 Luca Nizzardo
 Lucjan Hanzlik
 Luisa Siniscalchi
 Łukasz Chmielewski
 Maciej Obremski
 Madalina Bolboceanu

Mahimna Kelkar
Maria Eichlseder
María Naya-Plasencia
Marilyn George
Marios Georgiou
Mark Abspoel
Mark Simkin
Mark Zhandry
Markulf Kohlweiss
Marshall Ball
Marta Mularczyk
Martin Albrecht
Martin Hirt
Mary Wooters
Masayuki Abe
Matteo Campanelli
Matthias Fitz
Mia Filic
Michael Reichle
Michael Rosenberg
Michael Walter
Michele Orru
Miguel Ambrona
Mingyuan Wang
Miran Kim
Miruna Rosca
Miyako Ohkubo
Mohammad Hajiabadi
Mohammad Hossein Faghihi Sereshgi
Monosij Maitra
Morgan Shirley
Mridul Nandi
Muhammed F. Esgin
Mustafa Khairallah
Naomi Ephraim
Nathan Manohar
Naty Peter
Navid Alarnati
Ngoc Khanh Nguyen
Nicholas Spooner
Nicholas-Philip Brandt
Nico Döttling
Nicolas Resch
Nicolas Sendrier
Nikolaos Makriyannis
Nikolas Melissaris
Nils Fleischhacker
Nina Bindel
Nirvan Tyagi
Niv Gilboa
Noah Stephens-Davidowitz
Olivier Blazy
Olivier Bronchain
Omri Shmueli
Orfeas Stefanos Thyfronitis Litos
Orr Dunkelmann
Oxana Poburinnaya
Patrick Derbez
Patrick Longa
Patrick Towa
Paul Rösler
Paul Zimmermann
Peter Gazi
Peter Rindal
Philippe Langevin
Pierre Briaud
Pierre Meyer
Pierrick Gaudry
Pierrick Mèaux
Po-Chu Hsu
Prabhanjan Ananth
Prashant Vasudeval
Pratik Sarkar
Pratik Soni
Pratyay Mukherjee
Pratyush Mishra
Qian Li
Qiang Tang
Qipeng Liu
Quan Quan Tan
Rachit Garg
Radu Titiu
Rajeev Raghunath
Rajendra Kumar
Ran Cohen
Raymond K. Zhao
Riad Wahby
Rishab Goyal
Rishabh Bhadauria
Rishiraj Bhattacharyya
Ritam Bhaumik
Robi Pedersen

Rohit Chatterjee
Rolando La Placa
Roman Langrehr
Rongmao Chen
Rupeng Yang
Ruth Ng
Saba Eskandarian
Sabine Oechsner
Sahar Mazloom
Saikrishna Badrinarayanan
Sam Kim
Samir Hodzic
Sanjam Garg
Sayandeep Saha
Schuyler Rosefield
Semyon Novoselov
Serge Fehr
Shai Halevi
Shashank Agrawal
Sherman S. M. Chow
Shi Bai
Shifeng Sun
Shivam Bhasin
Shota Yamada
Shuai Han
Shuichi Katsumata
Siang Meng Sim
Somitra Sanadhya
Sonia Belaïd
Sophia Yakoubov
Srinivas Vivek
Srinivasan Raghuraman
Sruthi Sekar
Stefano Tessaro
Steve Lu
Steven Galbraith
Stjepan Picek
Sumegha Garg
Susumu Kiyoshima
Sven Maier
Takahiro Matsuda
Takashi Yamakawa
Tal Moran
Tamer Mour
Thom Wiggers
Thomas Agrikola
Thomas Attema
Thomas Debris-Alazard
Thomas Decru
Tiancheng Xie
Tim Beyne
Titouan Tanguy
Tommaso Gagliardoni
Varun Maram
Vassilis Zikas
Venkata Koppula
Vincent Zucca
Virginie Lallemand
Ward Beullens
Wei Dai
Willy Quach
Wouter Castryck
Xiao Liang
Xiao Wang
Xiong Fan
Yael Kalai
Yan Bo Ti
Yann Rotella
Yannick Seurin
Yaobin Shen
Yashvanth Kondi
Yfke Dulek
Yiannis Tselekounis
Yifan Song
Yilei Chen
Yixin Shen
Yongsoo Song
Yu Long Chen
Yu Sa
Yue Guo
Yuncong Hu
Yupeng Zhang
Yuriy Polyakov
Yuval Ishai
Zahra Jafargholi
Zeyong Li
Zhengfeng Ji
Zichen Gui
Zuoxia Yu
Zvika Brakerski

Contents – Part IV

Zero Knowledge

Witness Authenticating NIZKs and Applications	3
<i>Hanwen Feng and Qiang Tang</i>	
Towards a Unified Approach to Black-Box Constructions of Zero-Knowledge Proofs	34
<i>Xiao Liang and Omkant Pandey</i>	
Compressing Proofs of k -Out-Of- n Partial Knowledge	65
<i>Thomas Attema, Ronald Cramer, and Serge Fehr</i>	
Mac'n'Cheese: Zero-Knowledge Proofs for Boolean and Arithmetic Circuits with Nested Disjunctions	92
<i>Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl</i>	
Time- and Space-Efficient Arguments from Groups of Unknown Order	123
<i>Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni</i>	

Encryption++

Broadcast Encryption with Size $N^{1/3}$ and More from k -Lin	155
<i>Hoeteck Wee</i>	
Fine-Grained Secure Attribute-Based Encryption	179
<i>Yuyu Wang, Jiaxin Pan, and Yu Chen</i>	
Multi-input Quadratic Functional Encryption from Pairings	208
<i>Shweta Agrawal, Rishab Goyal, and Junichi Tomida</i>	
Functional Encryption for Turing Machines with Dynamic Bounded Collusion from LWE	239
<i>Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada</i>	
Receiver-Anonymity in Rerandomizable RCCA-Secure Cryptosystems Resolved	270
<i>Yi Wang, Rongmao Chen, Guomin Yang, Xinyi Huang, Baosheng Wang, and Moti Yung</i>	

Foundations

White Box Traitor Tracing 303
Mark Zhandry

Does Fiat-Shamir Require a Cryptographic Hash Function? 334
Yilei Chen, Alex Lombardi, Fermi Ma, and Willy Quach

Composition with Knowledge Assumptions 364
Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss

Non-interactive Batch Arguments for NP from Standard Assumptions 394
Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin

Targeted Lossy Functions and Applications 424
Willy Quach, Brent Waters, and Daniel Wichs

The t -wise Independence of Substitution-Permutation Networks 454
Tianren Liu, Stefano Tessaro, and Vinod Vaikuntanathan

Low-Complexity Cryptography

Low-Complexity Weak Pseudorandom Functions in $AC_0[MOD2]$ 487
Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl

MPC-Friendly Symmetric Cryptography from Alternating Moduli: Candidates, Protocols, and Applications 517
Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha

No Time to Hash: On Super-Efficient Entropy Accumulation 548
Yevgeniy Dodis, Siyao Guo, Noah Stephens-Davidowitz, and Zhiye Xie

Protocols

A Logarithmic Lower Bound for Oblivious RAM (for All Parameters) 579
Ilan Komargodski and Wei-Kai Lin

Oblivious RAM with *Worst-Case* Logarithmic Overhead 610
Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, and Elaine Shi

Puncturable Pseudorandom Sets and Private Information Retrieval with Near-Optimal Online Bandwidth and Time 641
Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce Maggs

Authenticated Key Exchange and Signatures with Tight Security
in the Standard Model 670
*Shuai Han, Tibor Jager, Eike Kiltz, Shengli Liu, Jiaxin Pan,
Doreen Riepel, and Sven Schäge*

KHAPE: Asymmetric PAKE from Key-Hiding Key Exchange 701
Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk

Author Index 731

Zero Knowledge



Witness Authenticating NIZKs and Applications

Hanwen Feng¹ and Qiang Tang²(✉)

¹ Beihang University, Beijing, China
feng_hanwen@buaa.edu.cn

² The University of Sydney, Sydney, Australia
qiang.tang@sydney.edu.au

Abstract. We initiate the study of witness authenticating NIZK proof systems (waNIZKs), in which one can use a witness w of a statement x to identify whether a valid proof for x is indeed generated using w . Such a new identification functionality enables more diverse applications, and it also puts new requirements on soundness that: (1) no adversary can generate a valid proof that will not be identified by any witness; (2) or forge a proof using her valid witness to frame others. To work around the obvious obstacle towards conventional zero-knowledgeness, we define entropic zero-knowledgeness that requires the proof to leak no partial information, if the witness has sufficient computational entropy.

We give a formal treatment of this new primitive. The modeling turns out to be quite involved and multiple subtle points arise and particular cares are required. We present general constructions from standard assumptions. We also demonstrate three applications in non-malleable (perfectly one-way) hash, group signatures with verifier-local revocations and plaintext-checkable public-key encryption. Our waNIZK provides a new tool to advance the state of the art in all these applications.

1 Introduction

Non-interactive zero-knowledge (NIZK) proof systems [8, 26] allow one to prove a statement by sending a single message to a verifier without revealing anything beyond the validity of the statement. NIZKs have been a ubiquitous tool in modern cryptography and play an essential role in constructing many important primitives such as chosen-ciphertext secure encryptions [35, 38], anonymous authentication tools such as group and ring signatures [20, 21], and many more.

While privacy is essential, some interesting functionalities become unattainable when considering the strong privacy definition where *all* partial information is protected. For example, doing a binary search for a plaintext in ciphertext is elusive when using a semantically secure encryption. How to construct secure schemes enabling certain functionalities, while maintaining the best possible privacy, is one of the central questions in modern cryptography and has been studied in a large amount of works in different contexts [5, 11, 14, 17, 32].

Part of the work was done while both authors were at New Jersey Institute of Technology.

In this paper, we turn our attention to NIZK proofs and consider to add an “identification” functionality: a witness w of a statement x (which potentially has many valid witnesses) in an NP language L can be used check whether a valid proof π showing $x \in L$ was generated by the witness w , i.e., $\text{Identify}(x, w, \pi) \stackrel{?}{=} 1$. It means that each witness w is “committed” to the proof π generated using w . Other than that, the proof will remain “zero-knowledge”. Such an exclusive checking capability immediately enables many interesting applications. For instance, one could easily realize a private/covert communication channel between administrators of an anonymous token system [31] as follows: administrators may consider using shared two witnesses w_1, w_2 to indicate whether a valid “anonymous certificate” falls into a certain blacklist (or whitelist) by using w_1 ; in this way, only the administrators obtain this extra information which remains hidden to everyone else in the system. As pointed out in the recent work of [31], such a tool is important to enable CDN providers to distinguish potentially malicious requests without breaching anonymity.

Adding this simple identification functionality also naturally posts new requirements on soundness: (1) if an attacker who knows a set of witnesses of a statement x generates a proof π for x , this proof must be identified by one of these witnesses; and (2) if a witness w is not known to an attacker (who may have other witnesses), any of the proofs generated by the attacker will not be identified by w , i.e., $\text{Identify}(x, w, \pi) = 0$.

We put forth a new notion called witness-authenticating NIZKs to capture all those requirements. Essentially, we add a way of distinguishing between different witnesses in NIZKs. As we will demonstrate soon in the applications, our new notion provides a new tool to advance the state of the art in multiple different domains: non-malleable (perfectly one-way) hash, group signature with verifier-local revocation, and plaintext-checkable public-key encryption.

1.1 Our Contributions

We overview our contributions in more detail below.

Definitional contributions. Adding a single identification functionality and defining the witness-authenticating NIZK proof system turn out to be highly involved; we have to revisit essentially every single property of the conventional NIZK proof system, and multiple subtleties exist.

Syntax and identifier witness. The basic idea is to augment a non-interactive proof system with an $\text{Identify}(\cdot)$ algorithm to check whether the witness he is possessing was used to generate the proof. However, often in practice, only a part of the witness (such as a secret key) is bound to a user; while other parts, such as random coins, may not be always available. To avoid unnecessary restrictions on the application, we introduce a generalization that we only require the Identify algorithm to take into a *part* of the witness. A bit more formally, we introduce a notion called *identifier witness*, which splits each witness w into an identifier witness w^I and a non-identifier witness w^{NI} . Using an identifier witness w^I of x , one can check whether a proof for x was generated using a witness in the form of (w^I, \star) . If $\text{Identify}(x, \pi, w^I) = 1$, we say π is authenticated by w^I . When

privacy is not considered in the context, we call such a proof system a witness-authenticating non-interactive proof system (waNIPS).

Entropic zero-knowledgeness. As a witness-authenticating proof has to convey at least a bit about the identifier witness that makes the identification functionality possible, the conventional zero-knowledgeness that hides all partial information of witness becomes out of reach. Therefore, we study the best possible privacy definition that we call the entropic zero-knowledgeness (entropic ZK), and call a waNIPS with this property a waNIZK.

- *Defining unpredictable sampler.* Similar to that semantic security is impossible for deterministic encryption, if an identifier witness can be guessed easily by the adversary, the Identify algorithm enables the adversary to trivially distinguish a real proof from a simulated proof. It follows that the privacy definition should be defined for languages with “unpredictable” (identifier) witnesses. To model that, we introduce an unpredictable sampler G which ensures that for a random sample $(x, w^I, w^{NI}) \leftarrow G(1^\lambda)$, given x , finding the associated identifier witness w^I is hard.

Several subtle issues appear. (1) In applications, if the whole statement is generated by the sampler, it may cause a trivial impossibility; for example, if a waNIZK is applied in a larger system, which requires an honestly generated public parameter pp (and the witness could be leaked completely if pp is malicious). We handle it by introducing a parameter generation algorithm that is not under the control of the adversary or sampler G . (2) In an adaptive setting, the sampler G could be generated by the attacker after seeing the CRS. But now, the sampled statement could simply contain one proof for which the corresponding witness is never output. This will enable a malicious prover to generate a proof without using any witness, which clearly violates the knowledge soundness. We get around this by requiring the unpredictability of the identifier witness to hold for *every* CRS value (instead of a randomly chosen one). Please see Sect. 2.1 for details.

- *Defining entropic ZK.* We define the entropic ZK, somewhat analogous to entropic security in encryptions [5], by capturing that adversaries still cannot learn anything more about w^I from π if w^I is sampled from the unpredictable sampler G (specified by the adversary). In conventional ZK, the whole witness is provided by the adversary; now adversary provides only a sampler. Directly integrating the unpredictable sampler to the conventional adaptive zero-knowledge definition would restrict adversary from learning side information about the witness via other or directly related proofs. We define another proof oracle to enable an adversary to obtain proofs on related statements. See Sect. 2.2 for details.

Soundness definitions. As very briefly mentioned above, soundness definitions also require a major upgrade because of the new identification functionality. Besides the conventional (knowledge) soundness, we require two new properties to show that the identifier witness to be “committed” to the proof, in the sense that 1) a proof must be identifiable by one of the identifier witnesses used in

the proof generation; 2) a malicious prover cannot “forge” a proof that will be identified by some identifier witness she does not know. Concretely,

- For the former property, we formalize it by augmenting the knowledge soundness (named *authenticating knowledge soundness*), saying that a witness extracted by a knowledge extractor from a valid proof not only validates the statement being proven, but also authenticates the proof.
- The latter property, which we call *unforgeability*, also relies on the unpredictable sampler; it is analogous to “unforgeability” in MAC. Namely, for a target witness generated from the unpredictable sampler, the adversary who can obtain multiple proofs generated from it still cannot produce a new proof that will be authenticated by this identifier witness.

Note that unforgeability defends against a malicious prover trying to “frame” a witness. In some applications, a malicious prover may generate a proof that links to a string which is not even a witness. We thus also introduce a notion called *identifier uniqueness*, which ensures that it is infeasible to generate a valid proof that could be authenticated by two different strings.

We remark that unforgeability and identifier uniqueness are *incomparable*: an attacker that cannot forge a proof being authenticated by an unknown witness may be able to produce one being authenticated by two witnesses he possesses; on the other hand, for technical reasons in the definitions, the identifier uniqueness is not strictly stronger either. But each could be useful in various applications when working together with other properties from the context.

There are several versions of weakening, e.g., in the CRS-independent setting; and strengthening. We refer detailed discussion in Sect. 2.3.¹

Constructions of witness-authenticating NIZK proofs. With the definitions and models settled, we are now ready to discuss the constructions.

Basic ideas. A natural idea of our waNIZK construction is to attach an authentication tag to the NIZK proof, and augment it with a proof of the validity of the tag. Verification could be easy, while security poses several challenges. Since we want to remain “zero-knowledge” when the witness is unpredictable, the tag should not leak any other partial information. I.e., it should be “simulatable”, even if the same witness is used to generate multiple proofs; further dealing with “unforgeability” incur extra difficulties in following different cases.

Warm-up constructions. Let us start with a special case where the identifier witness is uniform (or pseudorandom). For example, in group/ring signatures, the identifier witness is each user’s secret key. We notice that simulatability can

¹ We note that in the group signature of [2], a related notion called testable weak zero-knowledge (TwZK) was introduced as an attempt to add identification functionality. However, TwZK was only against uniform adversaries. Thus it can only be applied to more restricted languages (where the restrictions were informally described) and was impossible for non-uniform adversaries. Besides, soundness definition and provable constructions were not discussed.

be realized by pseudorandomness, and we could simply use the witness as the key to generate the tag using a PRF. Namely,

$$\text{Tag}_{\text{PRF}}(w^I) = (t, \text{PRF}(w^I, t)), \text{ for a random } t.$$

The “simulatability” and unforgeability of this tag simply follow the pseudorandomness, which in turn ensures the entropic zero-knowledgeness and unforgeability (the underlying NIZK should satisfy certain “non-malleability” to prevent from modifying a valid proof). If the identifier uniqueness is in need, we can further require the collision-resistance of the PRF [19]. We remark that this solution that enables very efficient instantiations, could already be useful.

A more general solution needs to deal with a general unpredictable sampler. We may apply a strong extractor [29] to the identifier witness to pump out a uniform key, then apply PRF to generate the tag. Some subtle issues arise immediately: (1) the same witness as a source may be used to generate multiple proofs (choosing different seeds). Thus, the extractor has to be re-usable thus requiring much more entropy (or the outer layer PRF needs to be related-key secure, which is only known for special relations); (2) a malicious prover might choose a “bad” seed to break the unforgeability, as the security of randomness extractor requires a uniform and independent seed. We resolve it by simply leveraging the common reference string, namely, using a part of CRS as the fixed seed. However, as a consequence, this technique can only be applied to the setting that the statements are from a *CRS-independent* sampler.

Full-fledged solution for CRS-dependent samplers. In many applications (e.g., in all three applications we will show), the unpredictable sampler may be generated after the adversary sees the CRS; thus, it depends on the CRS. But the construction now cannot simply obtain a string (e.g., the seed) from the CRS. Instead, we need to somehow “force” the honest behavior.

Let us examine the two soundness issues above: it is not clear how to force the same random seed to be used for every prover (if we do not want to get into the difficulty of reusable extractor or related-key secure PRF); moreover, proving a seed is generated uniformly already seems elusive. These obstacles motivate us to deviate from the Extract-then-PRF path. We first note that there are alternatives for “simulatability”. Also, to ensure the honest generation of randomness (such as seed) used in generating the tag, we may explore a parameter with structure or certain functionality so that we can prove and further bind the witness to the tag. Since we still need the identification function, those observations together lead us to the choice of deterministic public-key encryption (DPKE).

More precisely, let DEnc be the encryption algorithm of a DPKE scheme. We first generate a fresh public key pk , encrypt w^I to c under pk , and set (pk, c) as the tag. One can easily check whether w' is the encrypted message (identify here) w.r.t c by checking $\text{DEnc}(pk, w') \stackrel{?}{=} c$.

Now for entropic ZK, we note that the DPKE can provide simulation security if the message is unpredictable. More importantly, this needs to hold even facing multiple proofs on potentially related statements. Viewing the statements as auxiliary input on the identifier witness, we can obtain those from DPKE

with multi-user security with auxiliary inputs, which can be based on d -linear assumption [13]. Next, for soundness, and particularly unforgeability, we first need to ensure the well-formedness of pk . We can leverage the correctness of encryption and just prove a well-formedness of the ciphertext. Furthermore, “unforgeability” can be obtained by using a simulation-extractable NIZK proof.

We remark that our construction offers a framework that can have a hierarchy of instantiations. If we want the resulting waNIZK systems to have stronger (or weaker) property, we can instantiate the underlying NIZK correspondingly. For details, we refer to Sect. 3.2.

Applications. Our new abstraction of waNIZK can provide a tool for many interesting applications. Here we will showcase three non-trivial applications in hash functions, anonymous authentication revocation, and encryption in more detail. Each of them advances the state of the art in the corresponding topic. We believe there are many more applications which we leave for future exploration.

Non-malleable (perfectly one-way) hash from standard assumptions. Many works have been around trying to realize partial properties of random oracles, ideally, via standard assumptions. Perfectly one-way hash and non-malleable hash are two important primitives for this purpose, in settings that include Bellare-Rogaway encryption scheme [6], HMAC [27], and OAEP [10].

Perfectly one-way hash requires its (randomized) evaluation algorithm to hide all partial information of the pre-image, even with some auxiliary inputs, while providing a verification algorithm to check the correctness of evaluation. Non-malleable hash requires that one cannot “maul” a hash value into a related one even with some auxiliary information about the pre-image. Both of them also require collision resistance. Currently, perfectly one-way hash w.r.t general auxiliary inputs is only known to exist under a not-efficiently-falsifiable assumption [18], which contradicts the existence of iO [16]; while non-malleable hash are either from perfectly one-way hash [9] or in the random oracle model [3]. Given the recent progress [30] on iO, the mere existence of non-malleable hash or perfectly one way hash (with general auxiliary inputs) is still open.

We confirm the feasibility by presenting a new framework for non-malleable (perfectly one-way) hash functions from waNIZKs that can be based on the standard assumptions like the d -linear assumption. The starting point is to view the hash as a commitment that allows others to verify the committed value: it computationally determines an input and hides all partial information. This view inspires us to obtain a non-malleable (and perfectly one-way) hash by adding a proof of well-formedness of the commitment via waNIZKs where the input is set as the identifier witness. Perfect one-wayness comes from entropic ZK, collision resistance from identifier uniqueness, while non-malleability comes from (related-witness) unforgeability. For details, we refer to Sect. 4.1.

Auxiliary-input group signatures with verifier-local revocation. Group signatures [21] allow a user to sign a message on behalf of a group while hiding the signer’s identity. A major issue is the revocation of users whose membership should be cancelled without influencing others. In group signatures with verifier-local

revocation (VLR) [12], the signing procedure and the group public key will be independent of the revocation list, making this primitive appealing for systems providing attestation capabilities. Indeed, some instantiations of VLR group signatures such as the direct anonymous attestation scheme [14] have been already widely deployed in trusted platform modules (TPM) including Intel’s SGX.

Many works have shown these TPMs are vulnerable to “side channel” attacks by which attackers could learn partial information about the secret key. One approach to mitigate the threat is to employ leakage-resilient cryptographic schemes. However, existing VLR group signature schemes [11, 12, 14, 15, 32] do not provide any security guarantee when auxiliary information about secret key is leaked. We therefore study the problem of constructing leakage-resilient VLR group signature scheme, particularly, in the auxiliary-input model, the strongest model capturing one-time memory leakage.

Interestingly, a VLR group signature scheme necessarily relies on a secret-key-based tag generation which is identifiable (for revocation), unforgeable, and does not leak any partial information about the identity of the signer (for security). Existing constructions leverage either algebraic approaches [12, 14, 15, 32] or generic approaches such as PRFs [11] to realize the mechanism via “pseudo-randomness”, which will not hold anymore facing auxiliary-input leakage.

We solve this dilemma by using waNIZKs. Our idea is to simply replace the simulation sound NIZK in the folklore construction of group signatures (for proving knowledge of a group membership certificate) with our waNIZK.

Plaintext-checkable encryption in the standard model. Plaintext-checkable encryption (PCE) is a public-key encryption [17], allowing one to search encrypted data with plaintext. Compared with DPKE [5], a PCE could still be randomized and provides a stronger security ensuring two ciphertexts encrypting the same message are unlinkable. Besides a more fine-grained security notion, PCE has also been shown useful for constructing other primitives such as group signatures with verifier-local verification.

Existing constructions [17, 34] are mostly secure in the random oracle model. However, in several scenarios, including the application to VLR group signatures [17] and achieving CCA-security via Naor-Yung [35], we need to prove properties about the plaintext of a PCE ciphertext via NIZKs. Random oracles clearly become unfavorable. Attempts exist [17, 33, 34] for standard-model PCE, but unfortunately they only work for uniform message distributions. In most scenarios plaintext messages are unlikely uniformly distributed. It follows that designing a standard-model plaintext-checkable encryption scheme for biased message distributions is a natural question.

We also answer this question and present a general framework for plaintext-checkable encryption, from any standard-model IND-CPA secure PKE and waNIZKs. Our idea is simple: we first encrypt m with the PKE and then prove the ciphertext is well-formed by using waNIZKs and setting m as the identifier witness. This framework naturally gives standard-model instantiations. Moreover, the identifier witness in our full-fledged construction is only required to be unpredictable, which allows to remove the restriction on uniform messages.

Notations. Throughout the paper, we use λ for security parameter. For an NP language L , we let R_L denote its membership verification relation; $(x, w) \in R_L$ or $w \in R_L(x)$ denote that $R_L(x, w) = 1$, $R_L(x)$ denote the set of all witnesses of x , and L_n denote $L_n = L \cap \{0, 1\}^n$. We illustrate other notations and recall definitions of NIZKs and computational entropy in the full version.

2 Syntax and Security Models

As explained in the introduction, we consider a non-interactive proof system working for an NP language L , where a statement may have multiple witnesses. There is an extra mechanism **Identify**, such that anyone having a witness $w \in R_L(x)$ can efficiently check whether a proof π for $x \in L$ was generated using w . On the other hand, we require such mechanism to be robust, *i.e.*, anyone who does not know w cannot produce a valid proof for $x \in L$ that will be identified as generated from w . We call such a proof system a *witness-authenticating* non-interactive proof system (waNIPS), since now every proof essentially is authenticated by the corresponding witness. Though intuitive, formulating the new properties while adapting existing properties turns out to be involved.

Identifier witness. We first notice that the straightforward formulation of waNIPS, in which the extra identification algorithm **Identify** takes the *whole* witness, sometimes, limits the applications – some part of witness, such as the randomness (or other information) used for generating the proof, may not be functionally important or even be available, but are still required for the identification.

Consider a class of applications (including the non-malleable hash and plaintext-checkable PKE applications that we will present soon), in which we may just use the proof to carry a bit covertly that can be extracted by **Identify**. Now other users who may know the actual *secret* cannot figure out the randomness freshly generated; thus, they will not be able to run **Identify**. It is easy to see that the actual secret is necessary and sufficient for the identification purpose.

We thus consider the notion of identifier witness. Formally, for a statement $x \in L$, its witness $w = (w^I, w^{NI})$ consists of an identifier part w^I and a non-identifier part w^{NI} , where w^I will be explicitly specified by a relation R_L^I (called an identifier relation of L), $R_L^I((x, w^{NI}), w^I) = 1$, or $w^I \in R_L^I(x)$ for short. Now we only need the identifier witness for the identification algorithm.² Formally,

Definition 1 (waNIPS). Let L be an NP language, and R_L^I be an identifier relation of L . A waNIPS on (L, R_L^I) is defined by four efficient algorithms:

- $\sigma \leftarrow \text{Setup}(1^\lambda)$. The setup algorithm outputs a CRS σ .
- $\pi \leftarrow \text{Prove}(\sigma, x, w)$. The prover algorithm takes as inputs σ , an instance $x \in L$ with its witness $w \in R_L(x)$, and outputs a string π called a proof.
- $b \leftarrow \text{Verify}(\sigma, x, \pi)$. The verifier algorithm takes as inputs σ , an instance x and a proof π , and outputs either 1 accepting it or 0 rejecting it.
- $d \leftarrow \text{Identify}(\sigma, x, \pi, w^I)$. This algorithm takes as input a valid proof π for some $x \in L$ and a string w^I . It returns either 1 indicating π was generated by a witness in the form of (w^I, \star) , or 0 otherwise.

² We stress that the notion of identifier witness *does not put any restriction* on the languages that can be proved, as the non-identifier part can be empty. In this case, the identifier part is simply the whole witness.

The first three describe a non-interactive proof system for L . We say π is authenticated by w^I if $\text{Identify}(\sigma, x, \pi, w^I) = 1$.

Completeness of waNIPS could be easily defined by describing the identification functionality and the proving functionality over honestly generated proofs, which covers the standard completeness of non-interactive proof systems.

Definition 2 (Completeness of waNIPS). We say a waNIPS for (L, R_L^I) is complete, if for every $x \in L_\lambda$, $(w^I, w^{NI}) \in R_L(x)$, for $\sigma \leftarrow \text{Setup}(1^\lambda)$, $\pi \leftarrow \text{Prove}(\sigma, x, (w^I, w^{NI}))$, the following holds:

$$\Pr[\text{Verify}(\sigma, x, \pi) = 1 \wedge \text{Identify}(\sigma, x, \pi, w^I) = 1] = 1.$$

2.1 Defining Unpredictable Sampler

Incompatibility between identification and zero-knowledgeness. Before introducing the formal security definitions, we first clarify a basic question: when is a waNIZK meaningful? The question arises given that the identification functionality is clearly incompatible with the standard zero-knowledgeness.

As a concrete example, consider the range proof system where we use a NIZK to prove a committed integer value m w.r.t. a commitment com belongs to the range, say $(1, 20)$. Seeing such a proof, the adversary learns nothing about m except its range. However, if we use a waNIZK to support identification, then everyone can simply check all values in $(1, 20)$ and completely recover the value of m ! This simple example hints a trivial impossibility for conventional zero-knowledgeness of waNIZK, for the languages whose identifier witness can be easily guessed. Similar situation appears in other settings, e.g., encryption schemes equipped with a plaintext-search functionality [5].

It follows that we should focus on “hard” statements that one cannot guess the identifier witnesses easily. The notion that a statement is “hard” clearly cannot stand in the worst case if we are considering a non-uniform adversary, since its advice string may encode the witness already. We thus consider a distribution over a language such that for any efficient adversary, a random sample from this distribution is “hard”, and a waNIZK proof system is expected to work for languages admitting such “hard” distributions.

A natural way to describe a distribution is to specify an (adversarial) sampler G which is a non-uniform PPT algorithm and on input a security parameter outputs an element $x \in L_\lambda$ and its witness $(w^I, w^{NI}) \in R_L(x)$ ³. The unpredictability of this sampler can then be quantified by unpredictability entropy [29] of the identifier witness w^I . More precisely, G is k -unpredictable when $\mathbf{H}^{\text{unp}}(W^I|X) \geq k(\lambda)$, where (X, W^I, W^{NI}) is a joint random variable output by $G(1^\lambda)$. While such a formulation is simple, we find it unnecessarily restrictive in certain situations. We present a more general formulation below.

³ Note that in general, it is unclear how to generate a witness from a statement, so we let the sampler to output x, w^{NI} together with w^I , but we put no restrictions on them. In principle, x, w^{NI} could even be fixed by the attacker and hardcoded into G as long as an unpredictable w^I can be generated.

Modeling a more general unpredictable sampler. When applying our waNIZKs in a larger cryptosystem, the statement may involve system parameters that are not under the control of the adversary. This seemingly minor point is actually essential. A subtle issue is that letting the adversarial sampler to generate the whole statement sometimes makes it hard to enforce the unpredictability of witness. For example, consider a public-key encryption scheme and a simple language $L_{\text{Enc}} := \{(pk, c); (m, r) : c = \text{Enc}(pk, m; r)\}$ where m is the identifier witness. Let G_{pk} be the following sampler:

$$pk = pk^*, m \leftarrow M_\lambda, r \leftarrow_{\$} \{0, 1\}^\lambda, c = \text{Enc}(pk^*, m; r) : \\ \text{return } (x = (pk^*, c), w^I = m, w^{NI} = r),$$

where M_λ is a high-entropy message distribution. Is G_{pk} an unpredictable distribution? In general, the answer is no since the adversary might have the secret key sk of pk^* . However, simply excluding such a sampler is not the right choice. In typical applications (for example, in our application of plaintext checkable encryption, cf. Sect. 4.3), the public key is generated honestly and not under the control of the adversary. And the message distribution is specified by the adversary after seeing the public key.

This oddity arises due to that the larger system where a waNIZK is employed already requires some honestly generated parameter. To capture this intuition, we define a separate parameter generation as a PPT algorithm PG. We let the sampler algorithm to take as input the parameter pp generated by PG, asking the distribution conditioned on $\text{PG} = pp$ to be unpredictable. Note that PG is not a part of our waNIZK syntax, usually specified by the applications. We remark that this is optional (which could be empty if there is no PG in the application).

Modeling CRS-dependent unpredictability. As a waNIZK assumes a CRS, which is publicly available and usually generated once for all, in some scenarios, adversaries might be able to specify an unpredictable sampler after seeing the CRS. In this most general case, we allow the adversary and the sampler algorithm G to take CRS as an input.

One tricky issue exists when measuring the unpredictability of the output (particularly the identifier witness) of this CRS-dependent sampler: the statement itself could be containing auxiliary input of the identifier witness. An extreme example of this auxiliary input could be a valid proof; though the witness is still unpredictable to the adversary, such kind of auxiliary input destroys knowledge soundness. If a malicious prover simply outputs such a hardcoded proof, she generates a proof without knowing any witness!

More serious issues will occur at a new “unforgeability” property we will introduce. We will give a more detailed discussion when we present the soundness definitions (see Remark 4). To rule out those trivial “attacks”, we require the sampler to be unpredictable *for every* CRS. In this way, the hardcoded proof would be automatically ruled out, as there always exists one particular CRS such that an extractor knowing the corresponding trapdoor can recover the witness from the proof, which violates the unpredictability requirement.

Taking all above discussions into consideration, we present the formal definition of unpredictable samplers.

Definition 3 (Unpredictable sampler). *Let G be a sampler for (L, R_L^I) . We say G is k -unpredictable w.r.t. a trusted parameter generation procedure PG , if for every CRS σ in the range of $\text{Setup}(1^\lambda)$, it holds that*

$$\mathbf{H}^{\text{unp}}(W_\sigma^I | X_\sigma, PP) \geq k(\lambda),$$

where $PP = \text{PG}(1^\lambda)$ and $(X_\sigma, W_\sigma^I, W_\sigma^{NI}) \leftarrow G(PP, \sigma)$.

Clearly, the basic requirement is $k = \omega(\log \lambda)$.⁴ If we are considering CRS-independent samplers, G simply does not take as input the CRS σ .

2.2 Entropic Zero-Knowledgeness

We present a new definition of entropic zero-knowledgeness, ensuring that nothing else is leaked except the identification bit to attackers who know the exact identifier witness (to attackers who do not know the exact witness, actually the zero-knowledgeness remains). Or, to put it another way, to rule out the “trivial attacks” caused by the added identification functionality, we consider zero-knowledge property w.r.t unpredictable samplers. Since now the attacker does not know the witness, we need to give the attacker the capability to learn extra side information from other related proofs using the same witness, and this again should exclude the trivial impossibilities. Formally defining this new property requires care. We illustrate the intuition and the definition below.

Integrating the unpredictable sampler. Let us first recall the conventional zero-knowledge property: for any statement x along with its witness w , the procedure that generates a CRS σ and a valid proof π using (x, w, σ) , can be emulated by a simulator without using the witness. The adaptive counterpart allows the attacker to specify a statement after seeing the CRS.

Now the identifier witness w^I (along with (x, w^{NI})) is produced by an unpredictable sampler G , which is specified by the attacker. The prover (denoted as a prover oracle \mathcal{O}_{P1}) takes the tuple (x, w^I, w^{NI}) from G and the CRS as input and generates a proof. We want that this proof can be simulated via a simulator (denoted as \mathcal{O}_{S1}) without using the witness (w^I, w^{NI}) .

Allowing attackers to learn side information from related proofs. In the conventional zero-knowledge definition, since the attacker (distinguisher) is given the witness, just asking the simulator to emulate the proof is sufficient. While in our new definition, since the distinguisher does not have the exact witness, directly plugging in the unpredictable sampler to the zero-knowledge definition is too weak, in the sense that the prover only proves once. But in practical applications, this is not the case. For example, in group signatures, adversaries are allowed to obtain multiple signatures, possibly for different messages, from one user. To lift this restriction, we will allow the distinguisher to

⁴ We can also measure the unpredictability by HILL entropy [29]. On the one hand, it brings more restrictions on the languages to be proved; On the other hand, for samplers with sufficient HILL entropy we can give more efficient constructions which we explain in details in the full paper.

adaptively obtain multiple proofs, which could be generated from independently sampled statements. Also, seeing a statement x (whose identifier witness is w^I), the adversary can ask the prover to prove another related statement \bar{x} , which has the same identifier witness w^I .

Formally, we let the prover oracle \mathcal{O}_{P1} (or \mathcal{O}_{S1}) be stateful, and augment a pair of new oracles \mathcal{O}_{P2} and \mathcal{O}_{S2} , which, with access to the states of \mathcal{O}_{P1} and \mathcal{O}_{S1} , take as inputs an index (that specifies a previously sampled tuple) and an extended sampler EG. EG generates an extended statement \bar{x} (and corresponding non-identifier witness) seeing x , which is associated with the same w^I . However, an arbitrarily extended statement may leak the entire w^I although the original statement hides it. To rule out the trivial impossibility, we put a restriction on the extended statement w.r.t a w^I that it will not leak more information than the original statement, and thus w^I is still unpredictable.

Definition 4. We say EG is an admissible extended sampler w.r.t. G and PG, if there exists a PPT algorithm $\widetilde{\text{EG}}$, such that for every σ , and any non-uniform PPT \mathcal{A} , the following holds that $pp \leftarrow \text{PG}$, $(x, w^I, w^{NI}) \leftarrow G(\sigma, pp)$, $(\bar{x}, \bar{w}^{NI}) \leftarrow \text{EG}(pp, \sigma, x, w^I, w^{NI})$, $\tilde{x} \leftarrow \widetilde{\text{EG}}(pp, x)$, $\Pr[(w^I, \bar{w}^{NI}) \in R_L(\bar{x})] = 1$ and $|\Pr[\mathcal{A}(\sigma, pp, \bar{x}, w^I) = 1] - \Pr[\mathcal{A}(\sigma, pp, \tilde{x}, w^I) = 1]| \leq \text{negl}(\lambda)$, where the probability is taken over the coin tosses of PG, G , EG, $\widetilde{\text{EG}}$ and \mathcal{A} .

We are now ready to present the formal definition of entropic ZK.

Definition 5 (Entropic ZK). A waNIPS Π for (L, R_L^I) satisfies the (multi-theorem) entropic zero-knowledgeness w.r.t. a parameter generation procedure PG and a class of unpredictable samplers \mathcal{G} , if there is a PPT simulator $\{\text{SimSetup}, \text{SimProve}\}$, such that for every non-uniform PPT adversary \mathcal{A} , it holds that

$$\left| \Pr \left[\begin{array}{l} \sigma \leftarrow \text{Setup}(1^\lambda) \\ pp \leftarrow \text{PG}(1^\lambda); G \leftarrow \mathcal{A}(pp, \boxed{\sigma}) \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_{P1}, \mathcal{O}_{P2}}(\sigma, pp) \end{array} \right] - \Pr \left[\begin{array}{l} (\sigma, \tau) \leftarrow \text{SimSetup}(1^\lambda) \\ pp \leftarrow \text{PG}(1^\lambda); G \leftarrow \mathcal{A}(pp, \boxed{\sigma}) \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_{S1}, \mathcal{O}_{S2}}(\sigma, pp) \end{array} \right] \right| \leq \text{negl}(\lambda),$$

where the real prover oracles $\mathcal{O}_{P1}, \mathcal{O}_{P2}$ and the simulator oracles $\mathcal{O}_{S1}, \mathcal{O}_{S2}$ are defined in Fig. 1. The sampler G should belong to \mathcal{G} . EG shall be an admissible extended sampler w.r.t. PG and G (cf. Definition 4).

Remark 1. Entropic ZK for CRS-independent samplers can be easily obtained by removing the CRS (the boxed σ in Fig. 1) from the input of \mathcal{A} and G ; it also suffices in interesting applications and admits more efficient constructions. For detailed elaborations, we defer to the full version.

2.3 Soundness Definitions

The conventional (knowledge) soundness of non-interactive proof systems ensures that a prover that can generate a valid proof must possess a witness. In our setting with an extra identification functionality, we essentially require

$\overline{\mathcal{O}_{P1}(\sigma, pp)}$ $i + +;$ $(x_i, (w_i^I, w_i^{NI})) \leftarrow G(\overline{\sigma}, pp);$ $st \leftarrow st \cup (i, x_i, (w_i^I, w_i^{NI}));$ $\pi_i \leftarrow \text{Prove}(\sigma, x_i, w_i^I, w_i^{NI})$ $\mathbf{return} (x_i, \pi_i)$	$\overline{\mathcal{O}_{S1}(\sigma, \tau, pp)}$ $i + +;$ $(x_i, (w_i^I, w_i^{NI})) \leftarrow G(\overline{\sigma}, pp);$ $st \leftarrow st \cup (i, x_i, (w_i^I, w_i^{NI}));$ $\pi_i \leftarrow \text{SimProve}(\sigma, \tau, x_i)$ $\mathbf{return} (x_i, \pi_i)$
$\overline{\mathcal{O}_{P2}(\sigma, pp, x_i, \text{EG}, st)}$ $\text{Find}(i, x_i, (w_i^I, w_i^{NI})) \in st$ $(\bar{x}, \bar{w}^{NI}) \leftarrow \text{EG}(pp, \sigma, x_i, (w_i^I, w_i^{NI}))$ $\bar{\pi} \leftarrow \text{Prove}(\sigma, \bar{x}, w_i^I, \bar{w}^{NI})$ $\mathbf{return} (\bar{x}, \bar{\pi})$	$\overline{\mathcal{O}_{S2}(\sigma, \tau, pp, x_i, \text{EG}; st)}$ $\text{Find}(i, x_i, (w_i^I, w_i^{NI})) \in st$ $(\bar{x}, \bar{w}^{NI}) \leftarrow \text{EG}(pp, \sigma, x_i, (w_i^I, w_i^{NI}))$ $\bar{\pi} \leftarrow \text{SimProve}(\sigma, \tau, \bar{x})$ $\mathbf{return} (\bar{x}, \bar{\pi})$

Fig. 1. The oracles. \mathcal{O}_{P1} (resp. \mathcal{O}_{S1}) and \mathcal{O}_{P2} (resp. \mathcal{O}_{S2}) share the state st which is initialized to be \emptyset . The counter i is initialized to be 0.

the identifier witness to be “committed” to the proof. Naturally, the soundness property also needs to be upgraded. In particular, we would need to ensure that a used witness must be identifiable; and a malicious prover could not “forge” a proof that points to a witness that is not known to her. (1) The former property can be realized augmenting the conventional knowledge soundness such that: from a valid proof, a witness not only can be extracted but also is bound to the proof. (2) The latter mimics the binding property and models that an attacker has access to multiple witnesses for a statement but still cannot frame any others that hold another witness unknown to the attacker. We call it *unforgeability*. Formulating those notions turns out to be highly involved, especially when considering slightly more advanced notions. Formally,

Definition 6 (Authenticating knowledge soundness). *We say a waNIPS Π for (L, R_L^I) satisfies the authenticating knowledge soundness, if there exists a PPT extactor $(\text{Ext}_0, \text{Ext}_1)$, s.t., for any non-uniform PPT adversary \mathcal{A} , (1) the output of Ext_0 is computationally indistinguishable with the real CRS:*

$$|\Pr[(\sigma, \xi) \leftarrow \text{Ext}_0(1^\lambda) : 1 \leftarrow \mathcal{A}(\sigma)] - \Pr[\sigma \leftarrow \text{Setup}(1^\lambda) : 1 \leftarrow \mathcal{A}(\sigma)]| \leq \text{negl}(\lambda),$$

and (2) any valid proof must be authenticated by the extracted witness:

$$\Pr \left[\begin{array}{l} (\sigma, \xi) \leftarrow \text{Ext}_0(1^\lambda), (x, \pi) \leftarrow \mathcal{A}(\sigma), (w^I, w^{NI}) \leftarrow \text{Ext}_1(\sigma, \xi, x, \pi) : \\ \text{Verify}(\sigma, x, \pi) = 1 \wedge [(w^I, w^{NI}) \notin R_L(x) \vee \text{Identify}(\sigma, x, \pi, w^I) \neq 1] \end{array} \right] \leq \text{negl}(\lambda).$$

Remark 2. A weaker definition, which we call *authenticating soundness*, only requires the existence of such (w^I, w^{NI}) instead of that \mathcal{A} must know the witness. In some concrete applications of NIZKs such as signatures of knowledge [20],

the knowledge extraction procedure can be done by external primitives such as PKE. Thus, the NIZK does not have to be knowledge sound. The authenticating soundness will suffice for replacing authenticating knowledge soundness in similar cases. This notions will be formalized in the full paper.

Unforgeability. This property captures the “authenticity” that an adversary cannot forge a proof that will be authenticated by an identifier witness that the adversary does not know. Like our entropic ZK definition, we will leverage the unpredictable sampler for (L, R_L^I) to capture an unpredictable target witness. More importantly, we would like this to hold even if the adversary can adaptively obtain many proofs from witnesses unknown to her (the “forgery” thus should be a new proof) as she wishes. Note that this property indeed ensures that an adversary cannot simply “maul” a proof, and thus it (along with authenticating knowledge soundness) will suffice for many applications (such as non-malleable hash and VLR group signatures) which originally need a simulation-extractable NIZK for realizing non-malleability.⁵

Definition 7 (Unforgeability). *Let Π be a waNIPS for (L, R_L^I) . We say Π satisfies unforgeability w.r.t. PG and a collection of unpredictable samplers \mathcal{G} (cf. Definition 3), if for any non-uniform PPT adversary \mathcal{A} , it holds that*

$$\Pr \left[\begin{array}{l} pp \leftarrow \text{PG}(1^\lambda); \sigma \leftarrow \text{Setup}(1^\lambda); G \leftarrow \mathcal{A}(pp, \boxed{\sigma}); \\ (x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{P_1}, \mathcal{O}_{P_2}}(\sigma, pp) : (x^*, \pi^*) \notin \text{Hist} \\ \wedge \text{Verify}(\sigma, x^*, \pi^*) = 1 \wedge \exists w^I \in st, \text{Identify}(\sigma, x^*, \pi^*, w^I) = 1 \end{array} \right] \leq \text{negl}(\lambda),$$

where $G \in \mathcal{G}$, and $\mathcal{O}_{P_1}, \mathcal{O}_{P_2}$ are prover oracles specified in Fig. 1. **Hist** denotes the query-response history of \mathcal{O}_{P_1} and \mathcal{O}_{P_2} , and st denotes the set of identifier witnesses generated by all calls (made by \mathcal{A}) to \mathcal{O}_{P_1} .

Remark 3. The unforgeability could be weakened and is still useful; for example, for CRS-independent statements can be obtained by removing all boxed items above. On the other hand, in certain applications, we also need to strengthen the unforgeability: it is required that an adversary can neither frame the target identifier witness nor any identifier witness *related* to it. We term the strengthened definition by *related-witness unforgeability*, and show its application to non-malleable hash functions. Details will be given in the full version.

Remark 4. Recall that in the CRS-dependent sampler definition, we insist that the unpredictability holds for *every* CRS. One reason is that the unforgeability may not be achievable when unpredictability only holds for a randomly sampled CRS. Now we can give a concrete example. Assume L is an NP language and admits an unpredictable sampler G_L . We define an extended language L' that $x' = (x, y) \in L'$ iff $x \in L$, and a sampler $G_{L'}$ which on input a CRS σ , directly outputs (x, π) , where $(x, w^I, w^{NI}) \leftarrow G_L(1^\lambda)$ and $\pi \leftarrow \text{Prove}(\sigma, x, w^I, w^{NI})$. Given the entropic ZK of π , the identifier witness output by $G_{L'}$ is unpredictable. However, \mathcal{A} can directly output π to break the unforgeability.

⁵ Different from the conventional simulation soundness, where the adversary is given simulated proofs, here we provide real proofs, which will be needed in applications.

Identifier uniqueness. Next, we discuss a special property of identifier uniqueness, (like unique signatures), which is useful when handling a case that the attacker may output a proof that will be identified by a string that is not even a witness. In certain applications (e.g., in our application of plaintext-checkable encryption), the attacker may try to fool the identify algorithm used by others. Note that unforgeability does not address such an attack. The identifier uniqueness of a waNIPS says it is infeasible to produce a valid proof and two different identifier witnesses such that the proof is authenticated by both of them.

Definition 8 (Identifier uniqueness). *We say a waNIPS Π for (L, R_L^I) satisfies the identifier uniqueness, if any non-uniform PPT \mathcal{A} , it holds that*

$$\Pr \left[\begin{array}{l} \sigma \leftarrow \text{Setup}(1^\lambda); (x, \pi, w_1^I, w_2^I) \leftarrow \mathcal{A}(\sigma) : \text{Verify}(\sigma, x, \pi) = 1 \wedge \\ \text{Identify}(\sigma, x, \pi, w_1^I) = 1 \wedge \text{Identify}(\sigma, x, (\pi, w_2^I)) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

2.4 Definitions with Auxiliary Inputs

All definitions built upon samplers can be further strengthened by allowing adversaries to obtain other auxiliary information (beyond the statements) about the identifier witness. This strengthening will be useful when applying waNIZKs to applications with auxiliary inputs (e.g., in our applications of non-malleable hash and group signatures with verifier-local revocation). We formalize those by considering an enhanced sampler G , which outputs an auxiliary information z about w^I as well. Accordingly, the output of an admissible extended sampler EG shall be computationally indistinguishable with that of the associated $\bar{\text{EG}}$, even when the auxiliary information z is given to the distinguisher. In the strengthened definitions, the prover oracle \mathcal{O}_{P1} and the simulation oracle \mathcal{O}_{S1} will also return the auxiliary input z for the sampled w^I . Formal definitions appear in the full version.

On the one hand, the auxiliary-input entropic ZK and unforgeability clearly subsume the original definitions. On the other hand, considering auxiliary inputs does not seem to introduce any additional difficulty in constructing waNIZKs, since the statement itself is already an auxiliary information about the identifier witness. Thereafter, when we refer to entropic ZK and unforgeability, we mean the auxiliary-input counterparts.

3 Constructing Witness-Authenticating NIZKs

In this section, we present our general constructions for waNIZKs.

Basic challenges. A folklore approach for adding a new property to NIZKs is to add some “tag” and extend the statement being proved. For example, when transforming a NIZK to a knowledge-sound NIZK [38], one attaches the encryption of the witness to the proof, which enables the “extractability” by decrypting the ciphertext. Like this folklore, the main idea behind our constructions is also to attach an “identifiable” tag (and proof of validity) to a NIZK proof, *s.t.* it can be identified with the corresponding identifier witness. The challenge is that the tag has to satisfy several seemingly conflicting constraints.

- For “zero-knowledgeness”. The tag should not leak any information about the identifier witness except the bit to a verifier knowing the corresponding identifier witness. Particularly, a tag generated from an unpredictable w^I should be “simulatable” (without using w^I), even conditioned on the potential auxiliary information about w^I . Moreover, as the identifier witness may be used to prove multiple times, the “simulatability” shall be ensured across multiple tags from w^I .
- For *soundness*. First, we note that just for unforgeability, the tag generation should have a form of unforgeability. Namely, without the identifier witness w^I , a malicious prover cannot produce a tag that can be identified by w^I (even when it knows the statement). If we want the identifier uniqueness, it should be infeasible to find two identifier witnesses identifying one tag, which essentially requires a form of collision resistance. While for authenticating (knowledge) soundness, we will have to make sure the extracted witness is *exactly* the one used to generate the proof (comparing to the standard knowledge soundness, which only requires extracting *one* witness).

3.1 Warm-up Constructions

First, as a warm-up, we show how to easily build waNIZKs for distributions where the identifier witness is pseudorandom (conditioned on statements). This construction can be already useful in, e.g., group-oriented (*accountable*) authentications where users’ secret keys can be pseudorandom. We present a very simple construction from a NIZK and a PRF. We then show how to easily lift this construction to be secure for unpredictable distributions that are *independent of the CRS* by using randomness extractors.

PRF-based tag: a construction for pseudorandom identifier witnesses.

In many applications such as group-oriented anonymous authentication (*e.g.*, group signatures, ring signatures), the witness is usually a secret key. In this case, the identifier witness could be pseudorandom conditioned on all public information (*e.g.*, a public key can be a commitment to the identifier). As a natural idea to generate a simulatable tag is to create a tag that is also pseudorandom, we use the witness as a key to generate the tag using a PRF, *i.e.*,

$$\text{Tag}_{\text{PRF}}(w^I) \rightarrow (t, \text{PRF}(w^I, t)), \text{ for a random } t.$$

It is easy to verify that, when w^I is pseudorandom (with sufficient length and conditioned on all side information available to adversaries), $\text{PRF}(w^I, t)$ is pseudorandom for *any* t . Thus the tag $(t, \text{PRF}(w^I, t))$ is “simulatable” (for a random t) and unforgeable (for every t). Using such a tag generation mechanism, we can construct a simple waNIZK for a language L that admits a pseudorandom witness distribution. To identify whether the proof was generated by (w_*^I, \star) , one just checks $\text{PRF}(w_*^I, t) \stackrel{?}{=} \text{PRF}(w^I, t)$. Moreover, by further requiring the PRF function to be collision-resistant, we can also achieve *identifier uniqueness*. Formal construction and analysis will be presented in the full version.

Lifting via randomness extraction: a construction for general CRS-independent distributions. The above approach cannot be applied to general unpredictable witness distributions. A natural idea is to transform an identifier witness into a uniform string. Randomness extractors [4, 29] are such a tool for generating a nearly uniform string from a random variable with enough entropy (called *source*), with the help of a short uniformly random string called *seed*. A computational extractor [29] would also be applicable even if the witness distribution is only computationally unpredictable.

Several tricky issues remain: (1) For “*zero-knowledgeness*”, the attacker may obtain multiple proofs generated using w^I . Since the seed is randomly chosen, the attacker essentially forces the same witness to be re-used with multiple different seeds and then the resulting outputs are used as the PRF keys; Thus we will require a reusable extractor [23, 24] (or related-key secure PRF [1]). Unfortunately, there are only a few reusable (computational) extractor constructions, which either have entropy requirements on the source [23], or rely on non-standard assumptions [24]. In our setting, the witness distribution sometimes is only computationally unpredictable. The status of related-key secure PRF is neither promising as existing constructions only allow simple correlations. (2) For *soundness*, a malicious prover may not generate the seed honestly. In this case, we won’t have the properties of extractors, which could be devastating for unforgeability. To see this, let us view the inner product as the special Goldreich-Levin extractor, but the malicious prover will simply use all-0 string as the seed. Now every witness can be used to identify such proof!

Luckily, since we are working in the CRS model, we could simply let the CRS include one single piece of uniform seed. The tag can be generated as follows:

$$\text{Tag}_{\text{Ext-PRF}}(w^I) = (r, t, \text{PRF}(\text{Ext}(w^I, r), t)), \text{ for } r \text{ in CRS and a random } t.$$

Leveraging this tag generation mechanism, we can have a construction for a language L that is secure w.r.t. k -unpredictable distributions. Formal construction and analysis will be presented in the full version.

Unfortunately, once we do not have the luxury that the sampler is *independent* of the CRS, we will need new ideas for the challenges.

3.2 The Full-Fledged Construction for CRS-dependent Distributions

It is known that in general, a randomness extractor is secure only when the source is independent of the seed (otherwise, seeing the seed, there will always exist a source distribution that makes the first bit of the extractor output to be 1). Thus, the unpredictable statement distribution must be independent of the CRS in the above approach. However, in many applications, the statement (and the corresponding witness distribution) might depend on the CRS, e.g., all three applications we will present soon. It follows that we need a more general solution that can handle a CRS-dependent witness distribution.

A more flexible tag generation. It is not hard to see that for any tag generation function $f(\text{params}, w) = \tau$, if params is from CRS, the adversary can

always find a witness distribution that depends on `params` such that the output τ can be recognizable. However, in the above approach using extract-then-PRF, moving the seed out of CRS and letting prover generate it will put us back facing the challenges of malicious seed and reusability, as described above.

To circumvent such a dilemma, we first note that realizing simulatability and unforgeability does not have to be via pseudorandomness. For simulatability, another alternative is encryption primitives. For the ease of checking, we consider using deterministic public-key encryptions (DPKE) to generate a tag.⁶ Regarding the unforgeability, we note it can be realized by adding a simulation-extractable NIZK proof to the tag. As the NIZK is already a component of our waNIZK construction, we can make the tag unforgeable by enforcing the “collision resistance”. Let `DEnc` be the encryption algorithm of a DPKE scheme, and we illustrate the tag generation mechanism below.

$$\text{Tag}_{\text{DPKE}}(w^I) \rightarrow (pk, \text{DEnc}(pk, w^I)), \text{ for a random public key } pk. \quad (1)$$

Next, we examine the previous challenges more closely.

- For “zero-knowledge”, simulatability via pseudorandomness requires each output to be “independently” pseudorandom, thus requiring “reusability” in strong extractors. The latter is highly non-trivial as there is only a fixed amount of entropy available in the witness. While for ciphertext as output, however, we do not have to insist on a pseudorandom ciphertext distribution. Actually, “reusability” is trivial in standard public-key encryption schemes as each ciphertext is like an independent sample. Of course, in the setting of DPKE (when considering the multi-user security), things get more complicated as no private randomness is used for encryption; we also need to consider the auxiliary input of the witness. Fortunately, Brakerski and Segev’s d -linear based construction [13] can satisfy the auxiliary-input security and the multi-user security simultaneously.
- For *soundness*, it was difficult to deal with malicious (prover-generated) seeds in the extractor setting, as there is no way to prove a seed is *sampled uniformly*. Nevertheless, if the parameters have some algebraic structure or functional properties, we may be able to enforce those features (for unforgeability) instead of proving distributional properties. For example, the decryptability condition (correctness) is such a property, when using encryption. In more detail, a malicious prover may still want to choose a malformed pk , but now we can ask the prover to attach a proof of “goodness” of pk , simply attesting there *exists* a secret key. The perfect correctness of encryption requires that for *every* valid key pair pk, sk , and every message m , $\text{Dec}(sk, \text{DEnc}(pk, m)) = m$. This automatically implies that the encryption

⁶ Another potential tool could be perfectly one-way hash with auxiliary inputs [18]. Those are probabilistic functions that satisfy collision-resistance and hide all partial information about its input even under with auxiliary input. Unfortunately, such a strong primitive is only known to exist under a not-efficiently-falsifiable assumption [24]; thus, its existence is elusive. In fact, it even contradicts with a form of obfuscation [16]. We would like to have a construction that relies on standard assumptions.

function DEnc for each valid pk defines an injective function, i.e., for any $w_1 \neq w_2$, $\text{DEnc}(pk, w_1) \neq \text{DEnc}(pk, w_2)$. Moreover, it is indeed the case for the DPKE instantiation we chose in [13]. In this way, a malicious prover cannot evade the checking or frame other witness holders!

The construction. Let us firstly specify the building blocks we will use.

- A deterministic public-key encryption (DPKE) scheme $\Sigma_{\text{de}} = \{\text{K}_{\text{de}}, \text{E}_{\text{de}}, \text{D}_{\text{de}}\}$ (whose formal definition is recalled in the full paper). We assume w.l.o.g. that the plaintext space contains all identifier witnesses of L . Particularly, we require the DPKE to be perfectly correct and PRIV-IND-MU-secure with respect to 2^{-k} -hard-to-invert auxiliary inputs which captures the security when one message is encrypted under multiple keys and the auxiliary input about the message are available to adversaries. We assume w.l.o.g. that there is a relation $R_{L_{\text{de}}}$ s.t. a key pair (pk, sk) is valid iff $R_{L_{\text{de}}}(pk, sk) = 1$.
- A NIZK proof system $\Pi_{\text{zk}} = \{\text{S}_{\text{zk}}, \text{P}_{\text{zk}}, \text{V}_{\text{zk}}\}$ for an NP language

$$L_{\text{CD}} := \{(x, pk, c); (w^I, w^{NI}, sk) : (w^I, w^{NI}) \in R_L(x) \wedge w^I \in R_L^I(x) \wedge c = \text{E}_{\text{de}}(pk, w^I) \wedge R_{L_{\text{de}}}(pk, sk) = 1\}; \quad (2)$$

The full-fledged construction $\Pi_{\text{CD}} = \{\text{Setup}, \text{Prove}, \text{Verify}, \text{Identify}\}$ for an NP language L with identifier relation R_L^I is presented in Fig. 2.

Security analysis. The completeness directly follows the completeness of the underlying NIZK proof system Π_{zk} and of the DKPE scheme Σ_{de} . Particularly, under an honest pk , $c = \text{E}_{\text{de}}(pk, w^I)$ uniquely determines w^I and thus the proof will not be mis-identified by another identifier witness.

We claim the security of Π_{CD} in the following theorem and present here only a security sketch: the statement being proved by Π_{zk} is formed by (x, pk, c) . (1) the knowledge soundness of Π_{zk} ensures one can extract a w^I and $c = \text{E}_{\text{de}}(pk, w^I)$. By the description of **Identify**, these together imply the authenticating knowledge soundness. (2) When Π_{zk} is sound, the public key pk contained in a valid proof should be a valid public key, and thus (pk, c) determines a unique plaintext (as the identifier), which ensures the identifier uniqueness. (3) Moreover, as the DPKE is PRIV-IND-MU-secure with respect to 2^{-k} -hard-to-invert auxiliary inputs, the proof can achieve the entropic ZK.

Regarding the unforgeability, at a high level, we show a contradiction that a successful adversary \mathcal{A} against this property will give rise to a successful adversary \mathcal{B} that could recover messages from DPKE ciphertexts. Specifically, since Π_{zk} is a simulation-extractable NIZK, \mathcal{B} can answer all prover oracle queries via a “hybrid” prover algorithm which returns a “proof” formed by (pk, c, π_{zk}) where (pk, c) is an honest encryption of the identifier witness while π_{zk} is a simulated proof. Note that \mathcal{A} cannot distinguish the real prover oracle and the hybrid prover oracle. Next, \mathcal{A} will issue a challenge proof $(pk^*, c^*, \pi_{\text{zk}}^*)$, for a challenge statement x^* , satisfying $c^* = \text{E}_{\text{de}}(pk^*, w^I)$, and \mathcal{B} can further leverage the knowledge extractor of Π_{zk} to extract w^I , which is the plaintext of these deterministic encryptions, from π_{zk} .

Setup (1^λ)
$\sigma_{zk} \leftarrow S_{zk}(1^\lambda)$ // generate a CRS of the underlying NIZK return $\sigma = \sigma_{zk}$
<hr/> Prove ($\sigma, x, (w^I, w^{NI})$)
$(pk, sk) \leftarrow K_{de}(1^\lambda)$ // generate the public key and the secret key $c \leftarrow E_{de}(pk, w^I)$ // encrypt the identifier witness under pk $\pi_{zk} \leftarrow P_{zk}(\sigma_{zk}, (x, pk, c), (w^I, w^{NI}, sk))$ // prove $x \in L \wedge (pk, c)$ are well-formed return $\pi = (pk, c, \pi_{zk})$
<hr/> Verify (σ, x, π)
$b \leftarrow V_{zk}(\sigma_{zk}, (x, pk, c), \pi_{zk})$ // check the validity of the proof π_{zk} return b
<hr/> Identify (σ, x, π, w^I)
$c' \leftarrow E_{de}(pk, w^I)$ // encrypt the identifier witness under the public key if ($c = c'$) then return 1 else return 0

Fig. 2. The full-fledged construction.

Actually, our construction can satisfy the stronger related-witness unforgeability. Specifically, in the definition, a successful adversary will output (x^*, π^*, ϕ^*) such that $\pi^* = (pk^*, c^*, \pi_{zk}^*)$ is authenticated by $\phi^*(w^I)$, where ϕ^* is a transformation where all preimages can be efficiently found (the class of such transformations is formalized by Chen et al.[22] and will be recalled in the full paper). Note that the adversary \mathcal{B} can still leverage the attacker to recover messages from DPKE encryptions: all queries to the prover oracle can be simulated as before; after extracting $\phi(w^I)$ from the challenge proof π^* , \mathcal{B} just outputs one preimage of $\phi^*(w^I)$ which will equal to w^I with a non-negligible probability.

Due to the lack of space, we defer detailed proofs in the full version.

Theorem 1. *Let Π_{CD} be the construction in Fig. 2, and the following results hold:*

- Π_{CD} satisfies the authenticating (knowledge) soundness, if Π_{zk} satisfies the (knowledge) soundness;
- Π_{CD} satisfies the identifier uniqueness, if Π_{zk} is sound, and the DPKE satisfies perfect correctness;
- Π_{CD} satisfies the entropic ZK w.r.t. all k -unpredictable samplers, if Π_{zk} is zero-knowledge, and Σ_{de} is PRIV-IND-MU-secure with respect to 2^{-k} -hard-to-invert auxiliary inputs.⁷

⁷ A basic requirement is $k = \omega(\log \lambda)$ s.t. it is possible to have such a DPKE scheme.

- Π_{CD} satisfies the (related-witness) unforgeability w.r.t. all k -unpredictable samplers, if Π_{zk} is a simulation-extractable NIZK, and Σ_{de} is PRIV-IND-MU-secure with respect to 2^{-k} -hard-to-invert auxiliary inputs.

Sketch of instantiation. Since the underlying DPKE scheme Σ_{de} shall satisfy the perfect correctness and the PRIV-IND-MU-security with respect to hard-to-invert auxiliary inputs, the only candidate so far is Brakerski and Segev’s d -linear based construction [13]. Particularly, this construction allows 2^{-k} -hard-to-invert auxiliary inputs where $2^{-k} \leq \frac{\nu(\lambda)}{q^{2d}}$. Here, ν is a negligible function in λ , d can be 1 when considering the DDH assumption, and q is the order of the DDH group which is usually $2^{\Theta(\lambda)}$. Accordingly, if we set $\nu(\lambda) = 2^{-\omega(\log \lambda)}$, the admissible samplers of our waNIZK construction Π_{CD} should be k -unpredictable for some $k \geq 2 \log q + \omega(\log \lambda)$.⁸ Regarding the underlying simulation-extractable NIZK Π_{zk} for L_{CD} , we note it could be realized via simulation-extractable NIZKs for general NP languages. Particularly, adaptive NIZKs for general NP languages are known to exist under the RSA assumption [26] or the LWE assumption [37], and we can add simulation extractability to them using standard tools including one-way functions and public-key encryptions as noted in [38]. In addition, since the tag generation procedure is algebraic (and Groth-Sahai-friendly), we can leverage the (simulation-extractable) Groth-Sahai proof system [28] to instantiate Π_{zk} , if the statement $x \in L$ that we wish to prove is also Groth-Sahai-friendly.

4 Applications

We will present three different applications in (non-malleable) hash, (group) signature, and (plaintext-checkable) public key encryption respectively, and we will show how to advance the state of the art in each domain.

4.1 Non-malleable (Perfectly One-Way) Hash Functions from Standard Assumptions

Many efforts have been made formalizing meaningful cryptographic properties to realize random oracles. Perfectly one-way hash [18] and non-malleable hash functions [9] are notable examples. They are used to instantiate random oracles in e.g., Bellare-Rogaway encryption [6], HMAC [27], and OAEP [10] respectively. In particular, a perfectly one-way hash is a probabilistic function that requires the output to hide all partial information about the input (even with auxiliary information about the input), while still enabling the check of the validity of an evaluation. And non-malleable hash requires that one cannot “maul” a hash value into a related one even with some auxiliary information about the pre-image. Moreover, collision resistance is also required in both as it is necessary

⁸ In certain applications, we may be interested in the relation between $n = |w^I|$ and k of admissible samplers. Note that for any constant $0 < \mu \leq 1$, there exists a sufficient large polynomial n such that $n^\mu \geq 2 \log q + \omega(\log \lambda)$. Namely, k can be sublinear in n , and in this case given (X, Z, PP) finding W^I is sub-exponentially hard.

for many of their interesting applications, such as instantiating random oracles in Bellare-Rogaway encryption [6, 9].

Unfortunately, both perfectly one-way hash and non-malleable hash (with general auxiliary inputs) have no construction from any efficiently falsifiable assumption [3, 9, 18, 24].⁹ In particular, Boldyreva *et al.* [9] presented constructions of non-malleable hash from perfectly one-way hash functions [18] and simulation-extractable NIZKs [38], thus directly inherits the non-efficiently falsifiable assumption from [18]; Baecher *et al.* [3] showed another construction for non-malleable hash, but it requires a random oracle. Recall that the main motivation of non-malleable hash was to instantiate random oracles.

Note that the drawbacks the non-standard assumptions made by [18] have become much more serious: the assumption is known to contradict the existence of iO [16], while recent progress [30] demonstrated the feasibility of iO from some well-studied assumptions. The mere existence of such a non-malleable hash or perfectly one-way hash becomes unclear, and a basic question remains:

Does there exist a non-malleable (or perfectly one-way) hash function w.r.t. general auxiliary information from standard assumptions?

We solve both problems by using waNIZKs. Our framework could give concrete constructions for non-malleable *and* perfectly one-way hash functions with any sub-exponentially hard-to-invert auxiliary inputs, assuming only the standard assumptions like d -linear assumption. We directly construct a hash function that satisfies *perfect one-wayness*, *non-malleability* and *collision resistance* simultaneously. We simply name it non-malleable (perfectly one-way) hash.

Definition. A hash function \mathcal{H} is defined by a triple of PPT algorithms:

- $\text{HK}(1^\lambda)$. Generate a key hk of the hash function.
- $\text{H}(hk, s)$. On inputs a key hk and an input s output a hash value y .
- $\text{HVf}(hk, s, y)$. On inputs hk , s and y return a decision bit.

The correctness requires for any hk, s , it holds that $\text{HVf}(hk, s, \text{H}(hk, s)) = 1$. For security, the hash function \mathcal{H} is required to first satisfy:

- *Perfect one-wayness* w.r.t. ϵ -hard-to-invert auxiliary inputs. I.e., for any distribution $S = \{S_\lambda\}_{\lambda \in \mathbb{N}}$ and any hint function hint such that hint is ϵ -hard-to-invert w.r.t. S , and for any non-uniform PPT adversary \mathcal{A} , it holds that

$$\Pr \left[\begin{array}{l} hk \leftarrow \text{HK}(1^\lambda), s_0 \leftarrow S_\lambda, s_1 \leftarrow \{0, 1\}^{|s_0|}, b \leftarrow_s \{0, 1\}, \\ y \leftarrow \text{H}(hk, s_b), b' \leftarrow \mathcal{A}(hk, y, \text{hint}(hk, s_0)) : b = b' \end{array} \right] \leq \text{negl}(\lambda).$$

- *Collision resistance*. I.e., for any non-uniform PPT adversary \mathcal{A} ,

$$\Pr \left[\begin{array}{l} hk \leftarrow \text{HK}(1^\lambda), (s, s', y) \leftarrow \mathcal{A}(hk) : \\ s \neq s' \wedge \text{HVf}(hk, s, y) = \text{HVf}(hk, s', y) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

⁹ Under standard assumptions, the only existing perfectly one-way functions with auxiliary inputs [7] does not enjoy the collision resistance; and the only non-malleable hash (also given in [9]) is only secure against a very special class of auxiliary input.

For definition of non-malleability, we adopt it from [3] as this game-based definition is easier to use (than the simulation definition from [9]), and sufficient for all major applications including Bellare-Rogaway encryption [6], HMAC [27], and OAEP [10]. Informally, non-malleability requires that an adversary, seeing a hash value $y = H(hk, s)$ and an auxiliary input $\text{hint}(hk, s)$, cannot find another y^* whose pre-image is meaningfully *related to* s . The formal definition appears in the full paper. We note the “relation” between the pre-images is described a transformation set Φ , namely, s' is Φ -related to s if $s' = \phi(s)$ for some $\phi \in \Phi$. The non-malleability is defined w.r.t. a transformation set Φ rather than any transformation ϕ , since there exists some relation such as constant transformations, for which this definition is hopeless. In this work, we will adopt on transformations that have the so-called bounded root space (BRS) and samplable root space (SRS) (denoted by $\Phi_{\text{brs}}^{\text{SRS}}$) developed in [22], which are the currently most general yet achievable class (see the full paper).

Construction. Observe that non-malleable (perfectly one-way) hash has three security requirements and a verifier algorithm on each input-output pair. If we start just with perfect one-wayness (without the verifier algorithm) which hides all partial information, there are plenty of candidates; for example, a *commitment* scheme. For the remaining challenges of collision resistance and validity checking (while maintaining best possible privacy), our waNIZK becomes an immediate choice. For non-malleability, it can come from related-witness unforgeability. We define the evaluation as first committing to its input and then attaching a proof of the well-formedness of the commitment using our waNIZK proof!

More precisely, let $\text{COM} = \{\mathcal{K}_{\text{com}}, \mathcal{C}_{\text{com}}\}$ be a commitment scheme, and let $\Pi_{\text{wa}} = \{\mathcal{S}_{\text{wa}}, \mathcal{P}_{\text{wa}}, \mathcal{V}_{\text{wa}}, \mathcal{I}_{\text{wa}}\}$ be a WA-NIZK for an NP language $L_{\text{nm}} := \{(c, k_{\text{com}}); (s, r) : c = \mathcal{C}_{\text{com}}(k_{\text{com}}, s; r)\}$, in which s is the identifier witness. Here we require Π_{wa} to satisfy the identifier uniqueness, the entropic ZK and the related-witness unforgeability w.r.t. all $(-\log \epsilon)$ -unpredictable samplers and the transformation set $\Phi_{\text{brs}}^{\text{SRS}}$. We present the detailed description in Fig. 3.

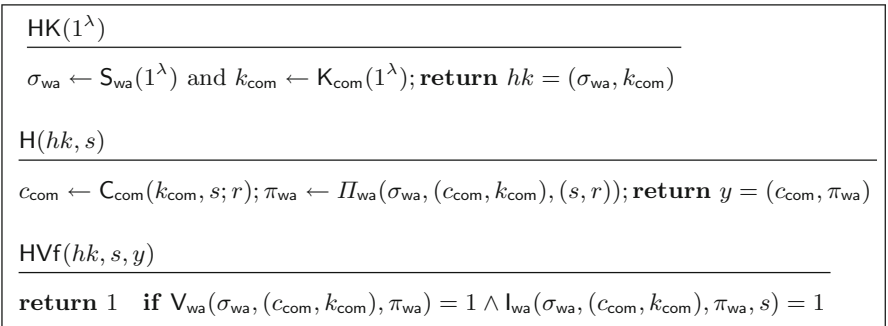


Fig. 3. Non-malleable Hash from commitment+ waNIZKs

Security analysis. The correctness follows the correctness of underlying primitives. Regarding collision resistance, if two distinct inputs (s_1, s_2) (which are

identifier witnesses) authenticate the sample proof, it immediately breaks identifier uniqueness. Notice that the hash value y consists of a hiding commitment and a WA-NIZK proof, both of which won't leak partial information about an unpredictable input. Thus, the perfect one-wayness follows easily. Regarding the non-malleability, notice that a mauled hash value must contain a mauled waNIZK proof, which is prevented by the related-witness unforgeability of the waNIZK. For detailed proofs, we defer them to the full version.

Theorem 2. *\mathcal{H} satisfies the perfect one-wayness w.r.t. ϵ -hard-to-invert auxiliary inputs, collision resistance, and non-malleability w.r.t. the transformation set $\Phi_{\text{brs}}^{\text{SRS}}$ and ϵ -hard-to-invert auxiliary inputs, if the commitment scheme COM satisfies computationally hiding, and Π_{wa} satisfies the identifier uniqueness, the entropic ZK and the related-witness unforgeability w.r.t. the transformation set $\Phi_{\text{brs}}^{\text{SRS}}$ and all $(-\log \epsilon)$ -unpredictable samplers.*

4.2 Group Signatures with Verifier-Local Revocation with Auxiliary Input

In group signatures with verifier local revocation (VLR) [12], we insist that the verifier can check by himself whether a signature is generated by a revoked group member, so that the group public key and the signing complexity are *independent* of revocation list which could be potentially long. In this section, we show how waNIZKs give rise to a simple VLR group signature scheme. Particularly, our construction enjoys auxiliary-input security, which is against a “side-channel” attacker who is allowed to see some computationally hard-to-invert function of the user’s secret key. To the best of our knowledge, known VLR group signatures cannot guarantee auxiliary-input security.

Why we consider the auxiliary-input security. Besides that “side-channel attacks” are a threat for every cryptographic primitive, and that the auxiliary-input model is currently the strongest model capturing memory leakage (more details about the model are referred to [25]), we find the auxiliary-input security for VLR group signatures is interesting both practice-wise and technical-wise.

Practice-wise, some instantiation of VLR group signatures, such as the direct anonymous attestation (DAA) [14] (along with its improved version, the EPID signature [15]), is adopted by the Trusted Computing Group as the standard for remote authentication, and implemented in several trusted platform modules (TPM) including Intel’s SGX. These TPMs are essential for computer security but are shown, by numerous works, vulnerable to side-channel attacks [36]. The study of auxiliary-input secure VLR group signature could enhance the security of TPMs against side-channel attacks.

Technique-wise, constructing auxiliary-input secure VLR group signatures turns out to be a non-trivial task. First, it is unclear how to easily “lift” existing constructions. Most of existing VLR group signature schemes (such as [12, 14, 15, 32]) leverage certain “pseudorandom functions” on a secret to preserve the anonymity while enable verifier-local checking. Such “pseudorandomness” either comes from underlying algebraic assumptions or directly from a PRF (e.g., a recent construction from Boneh et al. [11]). Unfortunately, with the auxiliary input on

the secret, “pseudorandomness” collapses. Essentially, in a VLR group signature, it will need an auxiliary-input secure secret-key-based tag generation mechanism that is identifiable (for realizing the revocation functionality), unforgeable, and does not leak any partial information about the signer identity (for anonymity). Our waNIZK provides a perfect tool.

The definitions. A VLR group signature scheme Σ_{gs} is defined by a tuple of three PPT algorithms.

- $\text{GS.KeyGen}(1^\lambda, n)$. It outputs a group public key gpk , and for each user $i \in [n]$, outputs the secret key $\mathbf{gsk}[i]$ along with the revocation token $\mathbf{grk}[i]$.
- $\text{GS.Sign}(gpk, \mathbf{gsk}[i], m)$. It outputs a valid signature ϑ for m under gpk .
- $\text{GS.Verify}(gpk, RL, \vartheta, m)$. It returns either 1 indicating that ϑ is a valid signature for m and was not signed by a revoked user whose token is in RL , or 0 otherwise. Here RL is a set of revocation tokens.

A VLR group signature scheme Σ_{gs} is correct, if for $(gpk, \mathbf{gsk}, \mathbf{grk}) \leftarrow \text{GS.KeyGen}(1^\lambda)$, every $RL \subset \mathbf{grk}$, every message $m \in \{0, 1\}^*$,

$$\text{GS.Verify}(gpk, RL, \text{GS.Sign}(gpk, \mathbf{gsk}[i], m), m) = 1 \Leftrightarrow \mathbf{grk}[i] \notin RL.$$

Note that this verification algorithm allows the group manager, who knows all revocation tokens, to *trace* signer’s identifier for every valid signatures. Specifically, if $\text{GS.Verify}(gpk, \emptyset, \vartheta, m) = 1$ and $\text{GS.Verify}(gpk, \mathbf{grk}[i^*], \vartheta, m) = 0$, the signer of ϑ will be traced to user i^* .

A VLR group signature scheme should satisfy the *anonymity* and the *traceability*. In the following, we briefly introduce the auxiliary-input counterparts of them, and the formal definitions are presented in the full paper. Particularly, we consider the auxiliary inputs as a hard-to-invert function on users’ secret keys along with the group public key, since user’s devices are much more vulnerable than the group manager’s device that is usually supposed to be well-protected.

- *Anonymity* ensures that the identity of an uncorrupted signer is indistinguishable from all possible signers, even when the adversary is allowed to see many signatures from all users and to corrupt some $\mathbf{gsk}[i]$ and $\mathbf{grk}[i]$. When considering the auxiliary-input anonymity, the adversary is further allowed to see a hard-to-invert function on (gpk, \mathbf{gsk}) .
- *Traceability* captures that any non-uniform PPT adversary \mathcal{A} can neither produce a valid signature-message pair (ϑ, m) that won’t be traced to any user, *i.e.*, $\text{GS.Verify}(gpk, \mathbf{grk}, \vartheta, m) = 1$, nor frame an uncorrupted user i^* , *i.e.*, $\text{GS.Verify}(gpk, \mathbf{grk}[i^*], \vartheta, m) = 0$, even when the adversary are allowed to obtain signatures from all users and to corrupt some $\mathbf{gsk}[i]$ and $\mathbf{grk}[i]$. When considering the auxiliary-input anonymity, the adversary is further allowed to see a hard-to-invert function on (gpk, \mathbf{gsk}) .

The construction. Our construction is based on the following observation. On rough terms, if a group signature scheme allows one to efficiently check whether a group signature was generated by using (a part of) $\mathbf{gsk}[i]$, then a VLR group signature can be built upon this as follows. 1) Set $\mathbf{grk}[i]$ to be (the specific part of) $\mathbf{gsk}[i]$, and 2) the verification algorithm performs as follows.

- Verify the group signature as the underlying verification algorithm does.
- If valid, for each $\mathbf{gsk}[i] \in RL$, identify whether ϑ was created by $\mathbf{gsk}[i]$. If ϑ is not identified by any $\mathbf{gsk}[i]$, accept it; otherwise, reject it.

The remaining part is to design a group signature with this identifiability. Note that the folklore of designing group signatures is to employ a simulation extractable NIZK to prove the knowledge of a group membership certificate [20] where the proof is taken as the signature. Then, our waNIZK will be an immediate choice to add the identifiability, by *replacing* the NIZK in this folklore. Moreover, the authenticating knowledge soundness and unforgeability of waNIZK would be enough to replace simulation extractability.

Specifically, we consider a pair (ID, Sig) , where ID is a bit string with sufficient length, and Sig is a digital signature for ID under a verification key vk_{sig} of the group manager. To sign a message m on behalf of the group, one just uses a waNIZK to prove the knowledge of such a pair w.r.t. (vk_{sig}, m) where ID is set to be the identifier. The auxiliary-input security follows the fact that all security guarantee of waNIZKs are preserved when auxiliary-information about witnesses¹⁰ is leaked to adversaries. More formally, let $\Sigma_{\text{sig}} = \{\mathbf{K}_{\text{sig}}, \mathbf{S}_{\text{sig}}, \mathbf{V}_{\text{sig}}\}$ be a standard digital signature scheme. Let $\Pi_{\text{wa}} = \{\mathbf{S}_{\text{wa}}, \mathbf{P}_{\text{wa}}, \mathbf{V}_{\text{wa}}, \mathbf{l}_{\text{wa}}\}$ be a waNIZK for the following language: $L_{\text{VLR}} : \{(vk_{\text{sig}}, m); (ID, \text{Sig}) : \mathbf{V}_{\text{sig}}(vk_{\text{sig}}, \text{Sig}, ID)\}$, where ID is the identifier witness. The formal description of the VLR group signature Σ_{gs} will be presented in the full paper.

The analysis. The correctness is easy to follow. Regarding the security, we first specify the admissible leakage function family \mathcal{F} . Assume the underlying waNIZK Π_{wa} satisfies the entropic ZK and the unforgeability w.r.t. all k -unpredictable samplers.

Definition 9. We say \mathcal{F} is admissible w.r.t. Σ_{gs} , if for every σ_{wa} in the range of $\mathbf{S}_{\text{wa}}(1^\lambda)$, and every $(vk_{\text{sig}}, sk_{\text{sig}})$ in the range of $\mathbf{K}_{\text{sig}}(1^\lambda)$, it holds that

$$\mathbf{H}^{\text{unp}}(ID|gpk = (\sigma_{\text{wa}}, vk_{\text{sig}}), f(gpk, \mathbf{S}_{\text{sig}}(vk_{\text{sig}}, sk_{\text{sig}}, ID), ID)) \geq k(\lambda),$$

where ID is a uniformly distributed random variable over $\{0, 1\}^{\text{id}(\lambda)}$ and id is an integer function polynomial in λ .

Note that the group public parameter gpk is independent of ID , and thus the admissible leakage function family \mathcal{F} is surely non-empty. Moreover, notice that the class of admissible leakage functions gets larger, if k is smaller.

Theorem 3. Let Σ_{sig} be a standard-model digital signature scheme satisfying EU-CMA security, Π_{wa} be a waNIZK for the language L_{VLR} that satisfies the authenticating knowledge soundness, the entropic ZK and unforgeability w.r.t all k -unpredictable samplers for L_{VLR} . Σ_{gs} is a secure VLR group signature scheme in terms of the auxiliary-input anonymity and the auxiliary-input traceability w.r.t. all admissible functions.

¹⁰ Since in the auxiliary-input model, this leakage could depend on the public parameter, which requires the underlying waNIZK to work for CRS-dependent samplers.

Proof (sketch). Recall that in the anonymity experiment, the goal of an adversary \mathcal{A} is to decide the signer’s identity for a signature that was generated by an uncorrupted user. The main idea of our proof is to show such a signature can be obtained by querying the prover oracles of Π_{wa} with an unpredictable sampler or an admissible extended sampler. By the definition of entropic ZK, a signature by an uncorrupted secret key (which is a proof π_{wa}) will not leak any useful information to adversaries beyond that its validity. The anonymity follows.

Regarding the auxiliary-input traceability, we note the adversary \mathcal{A} wins either when $(\text{GS.Verify}(gpk, \mathbf{grk}[i^*], \vartheta^*, m^*) = 0$ for the target user i^* , or when $(\text{GS.Verify}(gpk, \mathbf{grk}, \vartheta^*, m^*) = 1$. If \mathcal{A} wins via the first condition, we can show it contradicts the unforgeability of Π_{wa} by following similar arguments in the anonymity proof. For the second condition, the authenticating knowledge soundness of Π_{wa} ensures that for each valid proof π_{wa} , one can extract (ID, Sig) such that ID authenticates π_{wa} . Given the EU-CMA security of the digital signature scheme, the extracted ID must be one generated by GS.KeyGen and thus be contained in \mathbf{grk} , which contradicts the second condition. The detailed formal proof will be presented in the full paper. \square

4.3 Plaintext-Checkable Encryption in the Standard Model

Plaintext-checkable encryption (PCE) [17] is a public-key encryption primitive that allows us to search encrypted data with plaintext messages but still enables randomized encryption. Compared with deterministic public-key encryption (DPKE) [5], PCE aims to find a more fine-grained definition between the search functionality while preserving best possible security, particularly, it ensures two ciphertexts encrypting the same message are unlinkable (all partial information is still hidden when the plaintext is not known to the attacker). Moreover, it was also shown to be useful for group signatures with verifier-local revocation and backward unlinkability [12].

Existing constructions [17, 33, 34] are either relying on random oracles or only working for uniform message distributions.¹¹ In most scenarios, messages are from biased distributions. It is thus a natural question to consider PCE in the standard-model for non-uniform message distributions.¹² In this section, we answer this question and present a generic transformation from a PKE scheme to a PCE scheme, via a simple application of our waNIZK.

Definition. A PCE scheme enables everyone having a public key pk , a ciphertext c and a message m , to check whether m is the plaintext of c under pk . Formally, it consists of four algorithms: KeyGen , Enc , Dec , PCheck . While the first three algorithms describe a standard PKE scheme, the last algorithm is as follows:

¹¹ We note that the recent scheme [33] claimed security in the standard model for any high-entropy message distribution. However, their proofs still implicitly assume that the message distribution is uniform. We defer details to the full paper.

¹² The plain-text equality tester, presented in [39], seems close to a PCE. However, it can only check whether a ciphertext encrypts a pre-chosen target value m^* , while a PCE allows us to test for any plaintext publicly.

- $\text{PCheck}(pk, c, m)$. Outputs 1 indicating c is an encryption of m under pk , and 0 otherwise.

Correctness requires that for every λ and m , $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$, $c \leftarrow \text{Enc}(pk, m)$, $\Pr[\text{Dec}(sk, c) = 1 \wedge \text{PCheck}(pk, c, m) = 1] = 1$, where the probability is taken over coin tosses of KeyGen and Enc . We follow the definitions from [17]:

- *Checking completeness*: No efficient adversary can output a ciphertext which decrypts to a message that is refused by PCheck .
- *Checking soundness*: No efficient adversary can generate a ciphertext c and a plaintext m such that c cannot be decrypted to m but $\text{PCheck}(pk, c, m) = 1$.
- *k -unlinkability*: It is infeasible to decide whether two ciphertexts encrypt the same message, when the message is from a message distribution whose min-entropy is larger than k and the message is not available to adversaries.

Formal definitions are recalled in the full paper.

The construction. As a PCE scheme is a special PKE scheme that supports the plaintext-checking functionality while preserving the best-possible privacy, the idea behind our transformation is to attach a waNIZK proof that demonstrates the underlying PKE ciphertext is well-formed. More precisely, let $\Sigma_{\text{pke}} = \{K_{\text{pke}}, E_{\text{pke}}, D_{\text{pke}}\}$ be a PKE scheme, and let $\Pi_{\text{wa}} = \{S_{\text{wa}}, P_{\text{wa}}, V_{\text{wa}}, I_{\text{wa}}\}$ be a waNIZK for the following language: $L_{\text{PCE}} := \{(c, pk); (m, r) : c = E_{\text{pke}}(pk, m; r)\}$ where the message m is the identifier witness. To encrypt a message m , our PCE scheme first encrypts it using Σ_{pke} , and uses Π_{wa} to prove the ciphertext is well-formed, where the CRS for Π_{wa} is a part of the public key. Everyone can check whether a ciphertext $(c_{\text{pke}}, \pi_{\text{wa}})$ encrypts a particular message m by running the identification algorithm I_{wa} on π_{wa} and m . We defer the formal construction Σ to the full version.

The analysis. The correctness follows the correctness of the underlying primitives. Regarding the security, we establish the following result.

Theorem 4. *The PCE scheme Σ satisfies checking completeness, checking soundness, and k -unlinkability, if Σ_{pke} is an IND-CPA PKE scheme with perfect correctness, and the waNIZK Π_{wa} satisfies the entropic ZK w.r.t. all k -unpredictable samplers, the authenticating soundness, and the identifier uniqueness.*

Proof (sketch). The checking completeness follows the authentication soundness of Π_{wa} , and the checking soundness is implied by the identifier uniqueness of Π_{wa} . Regarding the k -unlinkability, we argue the distribution $G = \{(x = (c, pk), w^I = m, \perp) : m \leftarrow M_\lambda; c \leftarrow \text{Enc}(pk, m)\}$ for L_{PCE} is k -unpredictable w.r.t. an honest key generation $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$, if the min-entropy of M_λ is greater than k . We note given (c, pk) finding w^I is not necessarily 2^{-k} -hard. Indeed, we can define the following distribution $\bar{G} = \{(x = (c, pk), y, \perp) : m, y \leftarrow M_\lambda; c \leftarrow \text{Enc}(pk, m)\}$. Ensured by the IND-CPA security of the PKE scheme, \bar{G} is indistinguishable with G . As no side information about y is given, the probability of guessing y should be not greater than 2^{-k} . According to our

definition k -unpredictable distributions, G is such a distribution, enabling us to deploy a waNIZK that satisfies the entropic ZK w.r.t. k -unpredictable samplers. The above argument helps us to avoid requiring the sub-exponential hardness of the underlying PKE scheme. We defer the formal proof to the full version.

References


1. Abdalla, M., Benhamouda, F., Passelègue, A., Paterson, K.G.: Related-key security for pseudorandom functions beyond the linear barrier. *J. Cryptol.* **31**(4), 917–964 (2018)
2. Alamélou, Q., Blazy, O., Cauchie, S., Gaborit, P.: A code-based group signature scheme. *Des. Codes Cryptogr.* **82**(1–2), 469–493 (2017)
3. Baecher, P., Fischlin, M., Schröder, D.: Expedient non-malleability notions for hash functions. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 268–283. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_18
4. Barak, B., Dodis, Y., Krawczyk, H., Pereira, O., Pietrzak, K., Standaert, F.-X., Yu, Yu.: Leftover hash lemma, revisited. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_1
5. Bellare, M., Boldyreva, A., O’Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_30
6. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS, pp. 62–73. ACM (1993)
7. Bellare, M., Stepanovs, I.: Point-function obfuscation: a framework and generic constructions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 565–594. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_21
8. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112. ACM (1988)
9. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 524–541. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_31
10. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 210–225. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_14
11. Boneh, D., Eskandarian, S., Fisch, B.: Post-quantum EPID signatures from symmetric primitives. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 251–271. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_13
12. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: CCS, pp. 168–177. ACM (2004)
13. Brakerski, Z., Segev, G.: Better security for deterministic public-key encryption: the auxiliary-input setting. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 543–560. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_31
14. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: CCS, pp. 132–145. ACM (2004)

15. Brickell, E., Li, J.: Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. *Int. J. Inf. Priv. Secur. Integr.* **1**(1), 3–33 (2011)
16. Brzuska, C., Mittelbach, A.: Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014*. LNCS, vol. 8874, pp. 142–161. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_8
17. Canard, S., Fuchsbauer, G., Gouget, A., Laguillaumie, F.: Plaintext-checkable encryption. In: Dunkelman, O. (ed.) *CT-RSA 2012*. LNCS, vol. 7178, pp. 332–348. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_21
18. Canetti, R.: Towards realizing random oracles: hash functions that hide all partial information. In: Kaliski, B.S. (ed.) *CRYPTO 1997*. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052255>
19. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions (preliminary version). In: *STOC*, pp. 131–140. ACM (1998)
20. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) *CRYPTO 2006*. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_5
21. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
22. Chen, Yu., Qin, B., Zhang, J., Deng, Y., Chow, S.S.M.: Non-malleable functions and their applications. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) *PKC 2016*. LNCS, vol. 9615, pp. 386–416. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_15
23. Dachman-Soled, D., Gennaro, R., Krawczyk, H., Malkin, T.: Computational extractors and pseudorandomness. In: Cramer, R. (ed.) *TCC 2012*. LNCS, vol. 7194, pp. 383–403. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_22
24. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: *STOC*, pp. 621–630. ACM (2009)
25. Faust, S., Hazay, C., Nielsen, J.B., Nordholt, P.S., Zottarel, A.: Signature schemes secure against hard-to-invert leakage. In: Wang, X., Sako, K. (eds.) *ASIACRYPT 2012*. LNCS, vol. 7658, pp. 98–115. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_8
26. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: *FOCS*, pp. 308–317. IEEE Computer Society (1990)
27. Fischlin, M.: Security of NMAC and HMAC based on non-malleability. In: Malkin, T. (ed.) *CT-RSA 2008*. LNCS, vol. 4964, pp. 138–154. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79263-5_9
28. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
29. Hsiao, C.-Y., Lu, C.-J., Reyzin, L.: Conditional computational entropy, or toward separating pseudoentropy from compressibility. In: Naor, M. (ed.) *EUROCRYPT 2007*. LNCS, vol. 4515, pp. 169–186. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_10
30. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. *IACR Cryptol. ePrint Arch.* **2020**, 1003 (2020)

31. Kreuter, B., Lepoint, T., Orrù, M., Raykova, M.: Anonymous tokens with private metadata bit. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 308–336. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_11
32. Libert, B., Vergnaud, D.: Group signatures with verifier-local revocation and backward unlinkability in the standard model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 498–517. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_34
33. Ma, S., Huang, Q.: Plaintext-checkable encryption with unlink-CCA security in the standard model. In: Heng, S.-H., Lopez, J. (eds.) ISPEC 2019. LNCS, vol. 11879, pp. 3–19. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34339-2_1
34. Ma, S., Mu, Y., Susilo, W.: A generic scheme of plaintext-checkable database encryption. *Inf. Sci.* **429**, 88–101 (2018)
35. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437. ACM (1990)
36. Oleksenko, O., Trach, B., Krahn, R., Silberstein, M., Fetzer, C.: Varys: protecting SGX enclaves from practical side-channel attacks. In: USENIX Annual Technical Conference, pp. 227–240. USENIX Association (2018)
37. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 89–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_4
38. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_33
39. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: FOCS, pp. 600–611. IEEE Computer Society (2017)



Towards a Unified Approach to Black-Box Constructions of Zero-Knowledge Proofs

Xiao Liang^(✉)  and Omkant Pandey

Stony Brook University, Stony Brook, NY 11790, USA
{liang1,omkant}@cs.stonybrook.edu

Abstract. General-purpose zero-knowledge proofs for all NP languages greatly simplify secure protocol design. However, they inherently require the code of the underlying relation. If the relation contains black-box calls to a cryptographic function, the code of that function must be known to use the ZK proof, even if both the relation and the proof require only black-box access to the function. Rosulek (Crypto'12) shows that non-trivial proofs for even simple statements, such as membership in the range of a one-way function, require non-black-box access.

We propose an alternative approach to bypass Rosulek's impossibility result. Instead of asking for a ZK proof directly for the given one-way function f , we seek to construct a *new* one-way function F given only black-box access to f , and an associated ZK protocol for proving non-trivial statements, such as range membership, over its output. We say that F , along with its proof system, is a *proof-based* one-way function. We similarly define proof-based versions of other primitives, specifically pseudo-random generators and collision-resistant hash functions.

We show how to construct proof-based versions of each of the primitives mentioned above from their ordinary counterparts under mild but necessary restrictions over the input. More specifically,

- We first show that if the prover entirely chooses the input, then proof-based pseudo-random generators cannot be constructed from ordinary ones in a black-box manner, thus establishing that some restrictions over the input are necessary.
- We next present black-box constructions handling inputs of the form (x, r) where r is chosen uniformly by the verifier. This is similar to the restrictions in the widely used Goldreich-Levin theorem. The associated ZK proofs support range membership over the output as well as arbitrary predicates over prefixes of the input.

Our results open up the possibility that general-purpose ZK proofs for relations that require black-box access to the primitives above may be possible in the future without violating their black-box nature by instantiating them using proof-based primitives instead of ordinary ones.

Keywords: Zero-knowledge · Black-box · Separation

This material is based upon work supported in part by DARPA SIEVE Award HR00112020026, NSF grants 1907908 and 2028920, and a Cisco Research Award. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, DARPA, NSF, or Cisco.

1 Introduction

Zero-knowledge proofs (ZKPs) are a method to prove that a statement is true without revealing any additional knowledge [16]. A major achievement in cryptography has been the construction of ZKPs for NP-complete problems [15]. Since every NP relation can be efficiently reduced to any NP-complete relation [3, 29, 34], this yields a ZKP for all languages in NP. Due to this reason, ZKPs for NP-complete problems are often called *general-purpose* proofs. As evidenced by numerous follow up works, general-purpose proofs have been incredibly useful to the theory of cryptography.

Early constructions of general-purpose ZKPs required only *black-box* access to any one-way function (OWF), i.e., they used the given OWF as an *oracle*. A *black-box construction* of this kind thus depends only on the input/output behavior of the given cryptographic primitive. In particular, it is independent of the specific implementation or *code* of the primitive.

A black-box construction is often preferred over a non-black one due to its attractive properties. For example, it remains valid even if the primitive/oracle is based on a *physical* object such as a noisy-channel or tamper-proof hardware [4, 11, 43]. Also, its efficiency does not depend on the implementation details of the primitive, thus establishing that efficiency can be theoretically independent of the primitive’s code.

Unfortunately, general-purpose proofs are not suitable when seeking a black-box construction for some desired cryptographic task since they inherently require the full code of the underlying relation to perform the NP reduction. In other words, if the relation requires black-box access to a OWF, the code of the OWF must be known *even though* neither the ZKP nor the relation needs it. In fact, this has been the main reason for the non-black-box nature of many cryptographic constructions that are otherwise optimal. Analogous black-box constructions often require significant effort and technical innovation, as evidenced by the secure computation literature, e.g., [2, 6, 8–10, 17, 18, 20, 22, 26–28, 31, 32, 36, 39, 42].

In light of the above situation, it is tempting to imagine a “dream version” of general-purpose proofs where, if the underlying relation R requires black-box access to a cryptographic function f , say from a specified class such as the class of OWFs, then so should the general-purpose ZKP for proving membership in R . We informally refer to such relations as *black-box relations*. Such a result, if possible, would greatly simplify the task of future black-box constructions and potentially unify the diverse set of techniques that exist in this area.

As one might suspect, this dream version is too good to be true. In his beautiful work, Rosulek [41] rules out ZKPs for proving *membership in the range* of a OWF f given as an oracle. More specifically, assuming injective OWFs, Rosulek rules out (even honest-verifier) *witness-hiding* protocols [7] for the relation $R^f = \{(y, x) \mid y = f(x)\}$ where f is chosen from the class of all OWFs and provided as an oracle to the protocol.

In contrast to the negative result for OWFs, a large body of literature constructs so-called *black-box commit-and-prove* protocols [18, 19, 23, 27, 30, 33]. Informally speaking, a commit-and-prove protocol between a committer and a receiver ensures that at the end of the protocol, the committer is committed to some hidden value satisfying a pre-defined property. This primitive can be constructed with only black-box access to an ordinary commitment scheme which may originally not support any proofs whatsoever. In many situations, commit-and-prove protocols serve as a good substitute for ordinary commitments; moreover, their ability to support proofs over committed values makes them a great tool for constructing larger black-box protocols.

In hindsight, we can view black-box commit-and-prove protocols as an alternative to bypass the aforementioned negative result of [41]. That is, instead of constructing ZKP directly for every OWF, we ask the following indirect question:

Given only black-box access to a OWF f , can we construct a new OWF F and a ZKP system Π_F for proving membership in the range of F ?

Of course, we can ask for general properties instead of merely range-membership.

The idea is that F can be used as a substitute for f in any computation $C^{(\cdot)}$ that requires only black-box access to OWFs. More importantly, it gives hope that general-purpose black-box ZKPs for proving the correctness of computation $C^{(\cdot)}$ may be possible since the correctness of responses from F can be ensured using Π_F , all while requiring only black-box access to f . We remark that we do not obtain such a result for general computations in this work and merely point out that the existence of (F, Π_F) may open a path towards it.

We call the pair (F, Π_F) a *proof-based one-way function* (PB-OWF). Analogously, we consider proof-based versions of other primitives, specifically pseudo-random generators (PRGs) and collision-resistant hash functions (CRHFs). Motivated by the aforementioned possibility of a general-purpose proof system for black-box cryptographic computations $C^{(\cdot)}$, this paper initiates a study of black-box constructions of proof-based cryptographic primitives. We obtain a mix of both negative and positive results as outlined below.

1.1 Our Results

Given the existence of black-box commit-and-prove protocols, it is not unreasonable to expect that black-box proof-based versions of OWFs, PRGs, and CRHFs might also exist. Interestingly, the fact that these primitives are *deterministic* functions really separates them from commitments. The existence of their proof-based versions seems to depend on how we view the input, as discussed below.

Negative Results via Black-Box Separation. In common applications of non-interactive primitives such as OWFs and PRGs, the entire input is usually controlled by the evaluator of these functions. We show that *proof-based PRGs* where the input (i.e., the seed) is *entirely* chosen by the evaluator cannot be constructed in a black-box manner from an (ordinary) OWF chosen from the

class of all OWFs. Since PRGs can be constructed in a fully-black-box manner from OWFs [14, 21, 24], this separates proof-based PRGs from ordinary PRGs.

More specifically, black-box construction of a proof-based PRG from (ordinary) OWFs consists of a *deterministic* and efficient oracle algorithm $G^{(\cdot)}$, along with an efficient protocol, $\Pi_G^{(\cdot)} = \langle P^{(\cdot)}, V^{(\cdot)} \rangle$, of two interactive oracle machines.¹ For every OWF f , algorithm G^f should be a PRG, and protocol $\Pi^f = \langle P^f, V^f \rangle$ should be a ZKP system for the relation $R_G^f = \{(y, x) \mid \text{s.t. } y = G^f(x)\}$. Then, we show that a *fully-black-box* reduction [25, 40] from proof-based PRGs to ordinary OWFs does not exist if the prover chooses the entire seed.

The range-membership relation $R^f = \{(y, x) \mid y = f(x)\}$ ruled out in [41] is a special case of the aforementioned relation $R_G^f = \{(y, x) \mid \text{s.t. } y = G^f(x)\}$. In our terminology, Rosulek rules out a special type of proof-based OWF $(F^{(\cdot)}, \Pi_F^{(\cdot)})$ where F is just a “delegate” for the oracle OWF; i.e., it returns the oracle’s response when queried on the given input. This is captured in [41] by formally defining the notion of *functionally-black-box* (FBB) protocols. In contrast, the relation we consider can make polynomially many queries to the oracle on arbitrary inputs and compute over the responses to produce the output. We extend the notion of FBB protocols to formally capture these extensions.

In part due to these differences and our overall goals, our negative result is incomparable to that of Rosulek’s. While Rosulek rules out black-box proofs for range-membership for OWFs assuming injective OWFs, ours is only a black-box separation, albeit without any additional assumptions. A black-box separation is the best one can hope for in our setting since *non-black-box* constructions of proof-based OWFs that use the code of the oracle trivially exist.

Positive Results. We next investigate whether mild restrictions on the inputs can help bypass the black-box separation result. One option is to consider modifications along the lines of the Goldreich-Levin (GL) hardcore predicate [14], where one considers a OWF F constructed from any given OWF f on inputs of the form (x, r) . This makes it possible to show that predicate $\text{hc}(x, r) := \bigoplus_i (x_i \cdot r_i)$ is hardcore for the modified function $F(x, r) := r \parallel f(x)$ even though a hardcore predicate for arbitrary OWFs is still unknown. These changes to the function and the input do not seem to significantly affect the applicability of their result.

We adopt a similar approach to construct proof-based primitives. Continuing with OWFs as example, we seek to construct a proof-based OWF $F^{(\cdot)}$ which can be instantiated with only black-box access to any OWF f , and takes inputs of the form (x, r) . As in the GL setting, x will act as the “main input” chosen by the evaluator/prover, and r will be publicly accessible from the output of $F^f(x, r)$. However, in a crucial difference, r will be *chosen by the verifier* during the execution of ZKP Π_F^f . There are no other restrictions on any of the objects. Some remarks are in order.

¹ Note that the protocol is allowed to depend on $G^{(\cdot)}$ but not on the oracle which may be arbitrarily chosen later. The same holds for the relation R_G^f introduced next.

1. In light of our black-box separation result, it is essential to let the verifier choose r since no other restrictions are present. This means that the computation of $y = F^f(x, r)$ must be performed during the proof. We formalize this by modeling Π_F^f as a *secure two-party computation* protocol for evaluating the functionality that on inputs x and r from relevant parties, returns y . The ZK property is captured by requiring simulation-based security against malicious receivers; for soundness we only require that the honest verifier, with high probability, does not output a y^* that is not in the range. This is effectively a black-box ZKP for the relation $R_F^f(r) = \{(y, x) \mid \text{s.t. } y = F^f(x, r)\}$.²
2. The verifier must choose r from an unpredictable distribution such as the uniform distribution over sufficiently long strings, since otherwise, the soundness would be impossible as a cheating prover can simply guess r , bringing us back to the setting of the separation result.
3. Since r may be maliciously chosen by the verifier to violate the one-way property of F^f , we require that for *every string* r , the function defined by $F^f(\cdot, r)$ is one-way as long as f is one-way.

We follow the same approach for formally defining proof-based versions of PRGs and CRHFs. Having settled on a satisfactory definition, we present black-box constructions of the proof-based versions of OWFs (for range membership), as well as PRGs and CRHFs, directly from their ordinary counterparts.

Theorem 1 (Informal). *There is a fully black-box construction of proof-based primitive as described above for range-membership and two-party inputs of the form (x, r) , assuming that primitive exists, where primitive $\in \{\text{OWF, PRG, CRHF}\}$.*

At first glance, one may wonder whether black-box commit-and-prove protocols already yield proof-based OWFs. That is, the commit phase of such protocols can be viewed as a one-way function over the input (x, r) where x is the value to be committed and r is the randomness, the output y is the transcript of the commit-phase, and the proof-phase plays the role of associated ZKP. This approach does not really work since the commit-and-prove protocols merely *bind* the prover to a well-defined value x . They do not guarantee that w.h.p. every accepting transcript has a valid “preimage” (x, r) that maps to it. In contrast, the soundness of range-membership proofs of proof-based OWFs requires that w.h.p. a preimage must exist for the output accepted by the honest verifier. At a technical level, the black-box commit-and-prove protocols are based on cut-and-choose techniques that can only guarantee that the accepted value is *close* to an honestly generated value, which is insufficient to guarantee a preimage.

Extensions. We show that it is possible to construct a slightly more general proof-system than merely range-membership for each of our proof-based primitives. Continuing with the OWF example, we can construct a black-box proof-based OWF F^f such that for any predicate ϕ , the verifier learns a value y with

² For now, we only focus on range-membership proofs. The definitional approach is consistent with the commit-and-prove literature, although there are important differences since we are dealing with deterministic primitives.

the guarantee that there exists an input (x, r) such that: (1) $y = F^f(x, r)$ where r is chosen uniformly by the honest receiver, (2) $x = \alpha \|x'$, and (3) $\phi(\alpha) = 1$. That is, we can support any predicate (in fact, computation of any function) over a *prefix* of the preimage of the output. The ZKP system here depends on the code of ϕ but not that of f as before.

This extension is motivated by similar results for commit-and-prove which are quite useful in constructing larger black-box protocols [18, 30]. We achieve this by presenting a new construction which combines our ideas for range-membership with the “MPC-in-the-head” technique [27].

Due to space constraints, these extensions are formally described in the full version [35].

2 Technical Overview

2.1 Black-Box Separation

We first present a very brief overview of our black-box separation. A detailed overview is given in Sect. 4.2 after setting up necessary notation and definitions.

Let us first recall how Rosulek [41] rules out FBB constructions of honest-verifier witness-hiding (HVWH) protocols for the range-membership of OWFs, assuming injective OWFs exist.

The proof starts by assuming that such protocols exist. In particular, when instantiated with an injective OWF f , the protocol (P^f, V^f) is HVWH for $R^f = \{(y, x) \mid \text{s.t. } y = f(x)\}$. Since f is injective, for a pair $(x^*, y^* = f(x^*))$ remapping $f(x^*)$ to a value different from y^* will give us a new OWF f' whose range does not contain y^* anymore. Moreover, the verifier accepts in $\langle P^f(x^*, y^*), V^f(y^*) \rangle$ with roughly the same probability as in $\langle P^{f'}(x^*, y^*), V^{f'}(y^*) \rangle$. This is because the only opportunity to distinguish these two executions is when the verifier queries its oracle on x^* ; but this happens with negligible probability because of the HVWH property of the protocol. However, this contradicts the soundness: V 's oracle now becomes f' , and $\nexists x$ s.t. $(y^*, x) \in R^{f'}$.

It is unclear how to reuse the above technique to rule out PB-OWFs. As mentioned earlier, there are no restrictions on how the $F^{(\cdot)}$ part behaves. In particular, it is not guaranteed that F^f is injective even if f is injective. Thus, “carving out” a value from the range of f may not affect the range of F^f .

To derive the desired contradiction, we take a fundamentally different approach to construct f' . We first define a set Q^{Easy} , which consists of only the queries made by the receiver with “high” probability during the (honest) execution $\langle S^f(x^*, y^*), R^f(y^*) \rangle$. We then define f' by maintaining the same behavior as f on Q^{Easy} , and re-sampling all the remaining points uniformly at random. Note that the receiver will still accept with “high” probability even if we change its oracle to f' , because f' and f only differ at the points that are queried with “low” probability (i.e., the points outside Q^{Easy}). Now, the only thing left is to show that y^* is not in the range of $F^{f'}$. Unfortunately, due to the generality of $F^{(\cdot)}$, we do not know how to do that.

However, if we switch our focus to a PB-PRG G^f (instead of PB-OWF F^f), we can prove the following lemma which helps separate PB-PRGs from OWFs:

Lemma 1 (Informal). *If we start with a y^* in the range of G^f , y^* is still in the range of $G^{f'}$ with probability $0.5 \pm \text{negl}(\lambda)$, where $G^{(\cdot)}$ is a PRG when instantiated with any OWF f as its oracle.*

Let us show the intuition behind Lem. 1. Assume the lemma is false. In the pseudo-randomness game for G^f , we show how to identify the case $y^* \in \text{Range}(G^f)$ correctly, with probability noticeably better than 0.5, thus contradicting pseudo-randomness. To do that, the adversary simply estimates the probability that $y^* \in \text{Range}(G^{f'})$. We will show that, if this probability is noticeably far from 0.5, $\Pr[y^* \in \text{Range}(G^f)]$ is also noticeably far from 0.5.

Doing this successfully requires the adversary to know Q^{Easy} , which it does not. However, the adversary can run the HVZK simulator many times to get an estimate \tilde{Q}^{Easy} for the real Q^{Easy} . We will show that \tilde{Q}^{Easy} suffices for our proof. Note also that the adversary needs to perform exponential work when computing the probability that $y^* \in \text{Range}(G^{f'})$, even if it knows the set \tilde{Q}^{Easy} . However, it only makes *polynomially many* oracle queries (when executing the HVZK simulator), which suffices for proving the *fully*-black-box separation.

2.2 Proof-Based One-Way Functions (and PRGs)

Let us start by considering the following basic construction for PB-OWF (F^f, Π_F^f) over inputs of the form (x, r) . The construction is based on “cut-and-choose” techniques where, the sender queries the oracle f on “blocks” of x , and the receiver checks a size- t random subset (defined by r) of the responses. This method is not sound since it can only guarantee that the sender’s response is correct on most but not all blocks. We will handle this issue by introducing a new idea.

Basic Construction. PB-OWF (F^f, Π_F^f) handles inputs of the form (x, r) . F^f computes as follows:³

1. Parse x as (x_1, \dots, x_n) .
2. Interpret r as a size- t ($t < n$) subset of $[n]$, denoted by $\{b_1, \dots, b_t\}$.
3. Output $y = (y_1, \dots, y_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel r$, where $y_i = f(x_i)$ for all $i \in [n]$.

On input x to S^f and r to R^f , the execution $\langle S^f(x), R^f(r) \rangle$ is as follows:

1. S^f parses x as (x_1, \dots, x_n) , and computes (y_1, \dots, y_n) via its oracle access to f (i.e., $y_i = f(x_i)$). It sends (y_1, \dots, y_n) to the receiver.
2. R^f sends its input r to S^f . Same as in F^f , the r specifies a size- t subset $\{b_1, \dots, b_t\}$ of $[n]$. Recall that the honest receiver’s input r is random. In this case, $\{b_1, \dots, b_t\}$ is a *random* subset of $[n]$.
3. S^f sends $(x_{b_1}, \dots, x_{b_t})$, i.e., the x_i ’s whose indices are specified by r .
4. R^f checks (via its oracle access to f) if $y_{b_i} = f(x_{b_i})$ for all $i \in [t]$. If all the checks pass, R^f output $y = (y_1, \dots, y_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel r$.

³ We use Prover/Verifier and Sender/Receiver interchangeably since our ZKP is captured by considering a secure computation style definition for two parties.

Completeness is straightforward; furthermore $F^f(\cdot, r)$ is trivially one-way for every r since $t < n$ and f is a OWF. Let us first consider the *honest-verifier zero-knowledge* (HVZK) property of protocol Π_F^f .

Recall that the ZK property is defined via the ideal/real paradigm for secure computation, and requires simulation-security against malicious receivers. Thus, to prove the HVZK property, we need to show an ideal-world simulator Sim for the honest receiver. This is easy as the honest receiver will always use the given input r , which is uniformly distributed. More specifically, Sim works by sampling a uniform r by itself, sending the r to the ideal functionality, and receiving back the output $y = (y_1, \dots, y_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel r$. With this y , Sim can easily generate a simulated transcript that is identically distributed to the real one.

ZK Against Malicious Receivers. The above simulation strategy does not work for malicious receivers, because they may not use the given input r . Therefore, the simulator needs to somehow extract the candidate input r^* from the malicious receiver. However, the receiver will not give out its r^* until the sender/simulator sends the $\{y_i\}_{i \in [n]}$ values.

We point out that this issue cannot be fixed using standard methods such as requiring the receiver to commit to r and to open it later. This is because later, we will introduce a pre-image editing condition, and require that the sender's computation of F^f be consistent with this editing.

We therefore use a different idea. We modify the protocol to use a black-box commit-and-prove scheme $\Pi_{\text{ZKcnp}} = (\text{BBCom}, \text{BBProve})$ with ZK property. This scheme has a pair of simulators $(\text{Sim}_1, \text{Sim}_2)$ that can be used to simulate the receiver's view in the commit phase and the prove phase respectively. Our new Π_F^f is the same as before, except for the following changes:

- In Step 1, instead of sending y_i 's as before, the sender commits to them using BBCom. Formally, the sender sets $\nu = (y_1, \dots, y_n)$ and executes BBCom(ν) with the receiver.
- In Step 3, the sender sends both $\{x_{b_i}\}_{i \in [n]}$ and the value ν . It then proves using BBProve that this ν is indeed the value committed in BBCom.

As before, the receiver needs (y_1, \dots, y_n) to execute Step 4. Now, these values are contained in ν , and BBProve guarantees that the sender cannot change ν .

With these modifications, we can prove the ZK property for malicious receivers as follows. The simulator starts by running Sim_1 (the commit-phase simulator) with the malicious receiver R^{*f} . In this way, the simulator can go through Step 1 smoothly, without knowing the actual $\{y_i\}_{i \in [n]}$ values. Then, it will receive the r^* from R^{*f} . The simulator sends r^* to the ideal functionality and receives back $y = (y_1, \dots, y_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel r^*$. It sends $\nu = (y_1, \dots, y_n)$ and $(x_{b_1}, \dots, x_{b_t})$ to the receiver. Then, instead of executing BBProve, the simulator invokes Sim_2 to help itself go through the BBProve stage. It is easy to see that the ν and $\{x_{b_i}\}_{i \in [t]}$ sent by the simulator meet the consistency requirement in Step 4. Relying on the ZK property of Π_{ZKcnp} , one can formally prove that the simulation is done properly.

Soundness and Preimage Editing. As mentioned earlier, the “cut-and-choose” structure is not sufficient to guarantee the existence of a preimage. To see that, consider a malicious sender who picks an $i^* \in [n]$ at random, sets y_{i^*} to some value not in the range of f , or behaves honestly otherwise. This malicious sender can still make the honest receiver accept with non-negligible probability, even if t is as large as $n - 1$ (the upper bound for t to achieve any non-trivial ZK property). This is addressed by modifying the construction of F^f .

We start by noting that the “cut-and-choose” trick ensures that most of the y_i ’s are “good” (i.e., having preimages under f). For example, if t is a constant fraction of n , then the protocol ensures (except for negligible probability) that at most k of the y_i ’s are “bad”, where k is another constant fraction of n . Therefore, our idea is to extend the range of F^f to include all the images y that have $\leq k$ bad y_i ’s. More specifically, our new F^f works as follows. On input (x, r) , it still interprets r as $\{b_1, \dots, b_t\}$. But it will parse x as

$$x = (x_1, \dots, x_n) \parallel \underbrace{(p_1, y'_{p_1}), \dots, (p_k, y'_{p_k})}_{\beta},$$

where the $\{p_1, \dots, p_k\}$ form a size- k subset of $[n]$. The evaluation of $F^f(x, r)$ consists of two cases:

- **Non-Editing Case:** if $\{b_1, \dots, b_t\} \cap \{p_1, \dots, p_k\} \neq \emptyset$, then it computes y as before, ignoring the β part. That is, it outputs $y = (s_1, \dots, s_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel r$, where $s_i = y_i$ for all $i \in [n]$.
- **Editing Case:** if $\{b_1, \dots, b_t\} \cap \{p_1, \dots, p_k\} = \emptyset$, then at positions specified by p_i ’s, it replace y_{p_i} with y'_{p_i} . Namely, it outputs $y = (s_1, \dots, s_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel r$, where $s_i := \begin{cases} y'_i & i \in \{p_1, \dots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \dots, p_k\} \end{cases}$.

Let us explain how this editing technique resolves the soundness issue. Consider a y^* learned by the honest receiver with input r . As mentioned before, there are at most k y_i values (among those contained in y^*) that do not have preimages under f . These values can be expressed as $\{y_{p_1}^*, \dots, y_{p_k}^*\}$, i.e., their indices are $\{p_1, \dots, p_k\}$. Moreover, this set of bad indices does not overlap with the $\{b_1, \dots, b_t\}$ specified by r ; otherwise, the receiver would abort when performing the checks in Step 4. Therefore, by setting the β part to $(p_1, y_{p_1}^*), \dots, (p_k, y_{p_k}^*)$, we will obtain a valid preimage for y^* under our new $F^f(\cdot, r)$.

One may wonder whether a malicious sender can cheat by taking advantage of the **editing** case. However, since the honest receiver will use a random r , the set $\{b_1, \dots, b_t\}$ will always overlap with $\{p_1, \dots, p_k\}$ (except for negligible probability). That is, although we prove soundness by relying on the **editing** case, it almost never happens in a real execution. So, this will not give malicious senders any extra power.

We remark that the above preimage-editing idea is compatible with our technique for achieving (full) ZK. Now, the sender will append (y_1, \dots, y_n) and β to the committed value ν . Upon receiving R^f ’s challenge r , the sender computes

$\mathbf{s} = (s_1, \dots, s_n)$ according to the above definition of F^f . It sends both \mathbf{s} and $\{x_{b_1}, \dots, x_{b_t}\}$ to the receivers. Then, it runs **BBProve** to prove that it does the editing (or non-editing) honestly. Note that this statement can be expressed as a predicate on the values \mathbf{s} , \mathbf{r} , β , and $\{y_i\}_{i \in [n]}$, where the last two are committed in $\text{BBCom}(\nu)$. Since it does not involve the code of f , the protocol remains black-box in f . We provide more details in Sect. 5.2.

Proof-Based PRGs. Following the above paradigm, we also obtain a proof-based PRG by simply replacing the oracle OWF f with a PRG in the above PB-OWF construction. We provide a formal treatment in the full version [35].

2.3 Proof-Based Collision-Resistant Hash Functions

Recall that a PB-CRHF consists of a function H^h and a protocol Π_H^h such that for any CRHF h :

- For all \mathbf{r} , $H^h(\cdot, \mathbf{r})$ is a CRHF; **and**
- $\Pi_H^h = (S^h, R^h)$ is protocol satisfying similar completeness, soundness and ZK properties as for our PB-OWFs, but w.r.t. H^h .

Let us first try to reuse the idea from our PB-OWFs. On input (\mathbf{x}, \mathbf{r}) , the H^h first parses \mathbf{x} as $(x_1, \dots, x_n) \parallel \beta$, where the β has the same structure as before, for the purpose of preimage editing. It then generates $\{y_i\}_{i \in [n]}$ where $y_i = h(x_i)$, and outputs $\mathbf{y} = \mathbf{s} \parallel (x_{b_1}, \dots, x_{b_t}) \parallel \mathbf{r}$, where the value $\mathbf{s} = (s_1, \dots, s_n)$ is computed by editing $\{y_i\}$ (in the same way as for our PB-OWFs).

Since h is also a OWF, the H^h is surely one-way. However, it is not collision-resistant. To see that, recall that in the **non-editing** case, the β part is not used when computing $H^h(\mathbf{x}, \mathbf{r})$. This implies the following collision-finding attack. For a fix \mathbf{r} , the adversary first computes $\mathbf{y}^* = H^h(\mathbf{x}^*, \mathbf{r})$ with an \mathbf{x}^* whose β part does *not* trigger the **editing** condition. Then, it can easily find many preimages for \mathbf{y}^* by using different β 's, as long as they do not trigger the **editing** condition. Therefore, we need to come up with a new editing method that does not compromise collision resistance.

To do that, we modify H^h as follows. We sample a public string z and hardwire it in H^h . In this way, H_z^h can be viewed as a member of the public-coin collision-resistant hash *family* indexed by z , instead of a single CRHF. Then, we can think of \mathbf{x} as containing additionally two strings τ and μ . When evaluating $H_z^h(\mathbf{x}, \mathbf{r})$, we will perform the editing if $\{b_1, \dots, b_t\} \cap \{p_1, \dots, p_k\} = \emptyset$ and $\alpha \neq z$ and $h(\tau) = h(z)$. Moreover, we include the value $\mathbf{t} = h(\beta \parallel \tau \parallel \mu)$ in the output \mathbf{y} . Intuitively, this hash of β in \mathbf{y} prevents the adversary from constructing collisions using a different β .

We now explain how to perform editing in this setting. First, we will include in \mathbf{x} an additional value τ such that $\tau \neq z$ and $h(\tau) = h(z)$. This allows us to trigger the **editing** condition. With z sampled randomly, it is not hard to see that such a τ exists with overwhelming probability⁴. We can then set β as

⁴ This holds if the size of range of h is exponentially larger than its image space. It is also worth noting that τ does not need to be efficiently computable, because our soundness proof (or the editing technique) is only an existential argument.

before to ensure that the (x_1, \dots, x_n) part is “edited” properly. However, note that the y^* here contains additionally a \mathfrak{t}^* value. To handle this, we modify the construction of H_z^f slightly—We require that, when the **editing** condition is triggered, H_z^f sets $\mathfrak{t} = \mu$ in its output y . With this change, when performing editing, we can simply let μ equal the \mathfrak{t}^* . It is not hard to verify that this editing technique will lead to a valid preimage for y^* .

Finally, we remark that our real construction uses a Merkle tree for the prefix (x_1, \dots, x_n) of x . We only put the Merkle root in y , instead of the element-wise hash values described above. The soundness can be proved following essentially the same idea as above, except that we now “edit” the Merkle tree, which is done by extending the editing ideas to the tree setting. This allows us to compress a prefix of any length to a *fixed-length* string, such as 256 bits if using SHA256 for h . We refer the reader to Sect. 6 for a formal treatment of PB-CRHF.

2.4 Supporting Predicates

We discuss how to extend our constructions using “MPC-in-the-head” to additionally guarantee not only that the output learned by the receiver is in the range of the deterministic primitives, but also that the set of preimages contains one whose prefix satisfies some predicate ϕ .

Let us take a fresh look at the PB-OWF construction. It first parses the input as $x = \alpha \parallel \beta$. The β is for preimage editing; and the $\alpha = (x_1, \dots, x_n)$ can be regarded as a form of **Encoding** the prefix of x , i.e. $\text{Enc}(\alpha) = (x_1, \dots, x_n)$. Then, it computes $y_i = f(x_i)$ for all $i \in [n]$. Since this is mainly to introduce hardness (or one-wayness) to the final output, we can refer to this step as **Hardness Inducing**.

To support the proof of a predicate ϕ , we update the construction with new **Encoding** and **Hardness Inducing** methods. We first secret-share α to $([\alpha]_1, \dots, [\alpha]_n)$ using a verifiable secret sharing (VSS) scheme. This can be viewed as a new encoding method: $\text{Enc}(\alpha) = \text{VSS}(\alpha) = ([\alpha]_1, \dots, [\alpha]_n)$.

Next, we commit to these shares using Naor’s commitment [38], which can be built in black-box from the oracle OWF f . This can be thought of as a new **Hardness Inducing** method. Now, the output of F^f is of the following form:

$$F^f(x, r) = (\text{Com}([\alpha]_1), \dots, \text{Com}([\alpha]_n)) \parallel ([\alpha]_{b_1}, \dots, [\alpha]_{b_t}) \parallel r.$$

In the protocol Π_F^f , we additionally ask the sender to compute the value $\phi(\alpha)$ using the MPC-in-the-head technique. That is, the sender imagines n virtual parties $\{P_i\}_{i \in [n]}$, where P_i has $[\alpha]_i$ as its input. These n parties then execute a MPC protocol w.r.t. to the ideal functionality, which recovers α from the VSS shares, and outputs $\phi(\alpha)$ to each party. Let v_i denote the view of party i from the execution. The sender first commits to these views, and then opens some of them (picked by the receiver) for the receiver to check that the MPC for $\phi(\alpha)$ was performed honestly. In this way, the receiver not only learns $\phi(\alpha)$, but also believes that the sender did not cheat.

Finally, we make a few remarks:

- To achieve soundness, we also need to apply the preimage editing idea to the above construction.
- Both the **VSS** and **Com** require randomness, which can come from x . That is, we require that the x is long enough such that it also contains an η part (in addition to α and β). This η will provide the randomness for **VSS** and **Com**.
- The above approach applies directly to the **PB-PRG** and **PB-CRHF** constructions to make them support predicates on the α part of the preimage.

3 Preliminaries

Familiarity with basic cryptographic concepts such as ensembles, indistinguishability, and interactive Turing machines, etc. are assumed; we refer to [12, 13] for formal treatments of these. We also provide additional preliminaries in the full version [35].

Notations. We use “ \setminus ” to denote set difference. That is, for any two sets A and B , $A \setminus B := \{x : (x \in A) \wedge (x \notin B)\}$. The security parameter is denoted by λ . Symbols $\stackrel{c}{\approx}$, $\stackrel{s}{\approx}$ and $\stackrel{i.d.}{\approx}$ are used to denote computational, statistical, and perfect indistinguishability respectively; and $\text{negl}(\lambda)$ denotes negligible functions of λ . For a distribution D , $x \leftarrow D$ means that x is sampled according to D . Unless emphasized otherwise, we assume uniform distribution by default. We use $y \in D$ to mean that y is in the support of D . For a set S we overload the notation by using $x \leftarrow S$ to indicate that x is chosen uniformly at random from S . PPT denotes probabilistic polynomial time.

Let p be a predicate and D_1, D_2, \dots probability distributions, then the notation $\Pr [x_1 \leftarrow D_1; x_2 \leftarrow D_2; \dots : p(x_1, x_2, \dots)]$ denotes the probability that $p(x_1, x_2, \dots)$ holds after the ordered execution of the probabilistic assignments $x_1 \leftarrow D_1; x_2 \leftarrow D_2; \dots$. The notation $\{x_1 \leftarrow D_1; x_2 \leftarrow D_2; \dots : p(x_1, x_2, \dots)\}$ denotes the new probability distribution over $\{(x_1, x_2, \dots)\}$.

Black-Box Zero-Knowledge Commit-and-Prove. We need a zero-knowledge commit-and-prove protocol $\Pi_{\text{ZKC}_{\text{nP}}}$ with the following additional properties:

- it consists of two *separate* phases: a **Commit** phase **BBCom** and a **Prove** phase **BBProve**;
- the **Commit** phase itself constitutes a statistically-binding commitment scheme;
- for a public predicate $\phi(\cdot)$, the **Prove** phase constitutes a zero-knowledge argument for the value $\phi(x)$, where x is the value committed in **BBCom**;
- $\Pi_{\text{ZKC}_{\text{nP}}}$ can be constructed assuming only black-box access to OWFs.

A formal definition can be found in the full version [35]. There exist constructions satisfying the above requirements (e.g., [2, 18, 27]).

The “One-Oracle” Separation Technique. We first recall in Def. 1 the notion of *fully-black-box* reductions. We say that P cannot be obtained from Q in a fully-black-box way if there is no fully-black-box reduction from Q to P .

Definition 1 (Fully-Black-Box reductions [40]). *There exists a fully-black-box reduction from a primitive Q to a primitive P , if there exist PPT oracle machines G and S such that:*

- **Correctness:** *For every (possibly-inefficient) f that implements P , G^f implements Q ;*
- **Security:** *For every (possibly-inefficient) f that implements P and every (possibly-inefficient) machine \mathcal{A} , if \mathcal{A} breaks G^f (w.r.t. Q -security), then $S^{\mathcal{A},f}$ breaks f (w.r.t. P -security).*

A paradigm to rule out fully-black-box constructions is to design an oracle \mathcal{O} , and show that, relative to \mathcal{O} , primitive P exists but Q does not. A critical step in this proof is to construct an oracle machine $\mathcal{A}^{\mathcal{O}}$ that breaks the security of Q . We emphasize that \mathcal{A} is allowed to be *computationally unbounded*, as long as it only makes polynomially-many queries to \mathcal{O} (see e.g., [1, 25]). Our fully-black-box separation results in Sect. 4 will follow this paradigm.

4 The Impossibility Results

4.1 Meta-functionally Black-Box Constructions

Functionally Black-Box Protocols. To capture MPC protocols that “do not know” the code of the target function g , Rosulek [41] proposes the following notion of functionally-black-box protocols.

Definition 2 (Functionally-Black-Box Protocols [41]). *Let \mathcal{C} be a class of functions, and let $\mathcal{F}^{(\cdot)}$ be an ideal functionality that is an (uninstantiated) oracle machine. Let $A^{(\cdot)}$ and $B^{(\cdot)}$ be PPT interactive oracle machines. Then, we say that $(A^{(\cdot)}, B^{(\cdot)})$ is a functionally-black-box (FBB) protocol for $\mathcal{F}^{\mathcal{C}}$ in a certain security model if, for all $g \in \mathcal{C}$, the protocol (A^g, B^g) is a secure protocol (in the model in question) for the ideal functionality \mathcal{F}^g .*

By instantiating \mathcal{C} and $\mathcal{F}^{(\cdot)}$ properly, Def. 2 could capture black-box constructions of many useful cryptographic protocols. For example, let \mathcal{C}_{OWF} be the collection of OWFs. For any $g \in \mathcal{C}_{\text{OWF}}$, let $\mathcal{F}_{\text{ZK}}^g$ be the functionality that takes input x from party A , queries its oracle g to obtain $y = g(x)$, and outputs y to party B . Such an $\mathcal{F}_{\text{ZK}}^g$ is essentially a zero-knowledge argument (of knowledge) functionality for statements of the form “ $\exists x$ s.t. $g(x) = y$ ”. However, Rosulek showed that if injective OWFs exist, then it is impossible to have FBB protocols that implement $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}$ with semi-honest security (in the standard MPC setting), even in the presence of an arbitrary trusted setup. Given the broad application of ZK proofs, this result is quite discouraging.

Meta-FBB Functionalities. Observe that the above $\mathcal{F}_{\text{ZK}}^g$ functionality simply collects input x from A , queries its oracle g , and sends $g(x)$ to B . It only plays the role of a delegate for A and B to interact with the OWF g . Therefore, it is tempting to investigate whether we can circumvent Rosulek’s lower bound by allowing the “delegate” \mathcal{F}_{ZK} to perform extra computations, such as preprocessing x , post-processing $g(x)$, or making multiple queries to the oracle g , etc.

More formally, we want a non-cryptographic and deterministic computation F (used to capture the aforementioned extra computations), such that $\mathcal{C}'_{\text{OWF}} = \{F^g \mid g \in \mathcal{C}_{\text{OWF}}\}$ is a collection of OWFs. And we hope that there exists a FBB protocol (A^g, B^g) implementing $\mathcal{F}_{\text{ZK}}^{F^g}$ for all $F^g \in \mathcal{C}'_{\text{OWF}}$ (we can also denote it as $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$). Note that we require $(A^{(\cdot)}, B^{(\cdot)})$ to access g in a black-box way only; they can make use the code of F . Since $\mathcal{C}'_{\text{OWF}}$ is also a collection of one-way families, $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ can be used as a substitute for $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}$, with the only overhead coming from the computations represented by $F^{(\cdot)}$. Because $F^{(\cdot)}$ is supposed to contain only simple non-cryptographic operations, the implementation of $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ should be as efficient as that of $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}$. Therefore, if this approach is possible, it will alleviate the negative implications of Rosulek's lower bound.

We can also interpret $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ as a new FBB functionality $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}[F]$, i.e., a new oracle machine $\mathcal{F}_{\text{ZK}}^{(\cdot)}[F]$ to be instantiated with oracle OWFs from the original collection \mathcal{C}_{OWF} . For any $g \in \mathcal{C}_{\text{OWF}}$, $\mathcal{F}_{\text{ZK}}^g[F]$ collects the input X from Party A , evaluates $F^g(X)$, and sends $y = F^g(X)$ to Party B .

With this interpretation, $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ is just an instantiation of Def. 2 with $\mathcal{F}^{(\cdot)} = \mathcal{F}_{\text{ZK}}^{(\cdot)}[F]$ and $\mathcal{C} = \mathcal{C}_{\text{OWF}}$. To distinguish with Rosulek's $\mathcal{F}_{\text{ZK}}^{(\cdot)}$ functionality. We call $\mathcal{F}_{\text{ZK}}^{(\cdot)}[F]$ the *Meta-FBB ZK Functionality*. Similarly, one can also extend other FBB functionalities in [41] (e.g., 2-party secure function evaluation $\mathcal{F}_{\text{SFE}}^{(\cdot)}$, pseudo-random generator $\mathcal{F}_{\text{PRG}}^{(\cdot)}$, where sender A holds the seed and receiver B holds the key) to the corresponding Meta-FBB version.

4.2 The Main Theorem

In this part, we show that although we relax Rosulek's FBB notion to the Meta-FBB one, there still exists strong impossibility result. More specifically, we prove that, given only black-box access to OWFs, it is impossible to build a PRG that admits Meta-FBB honest-verifier zero-knowledge protocols.

Definition 3 (Fully-Black-Box PRGs from OWFs). *Let \mathcal{C} be the collection of OWFs. A (deterministic) polynomial-time oracle machines $G^{(\cdot)}$ is a fully-black-box construction of PRG from OWF if there exists a PPT oracle machines $A^{(\cdot)}$ such that:*

- **Correctness:** $\forall f \in \mathcal{C}$, G^f is a PRG;
- **Security:** $\forall f \in \mathcal{C}$ and every (possibly inefficient) machine M , if M breaks the pseudo-randomness of G^f , then $A^{M,f}$ breaks the one-wayness of f .

Theorem 2 (Main Theorem). *Let $\mathcal{C} = \{f \mid f \text{ is a OWF}\}$. There does not exist a (deterministic) oracle machine $G^{(\cdot)}$ such that*

1. $G^{(\cdot)}$ is a fully-black-box construction of PRG from OWF; **and**
2. for all $f \in \mathcal{C}$, there exists a stand-alone, Meta-FBB, honest-verifier zero-knowledge argument system $\Pi^f = \langle P^f, V^f \rangle$ for the functionality $\mathcal{F}_{\text{ZK}}^f[G]$.

Before showing the full proof in Sect. 4.3, let us provide the high-level idea.

Proof Sketch. We start by assuming (for contradiction) that the $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ specified in the theorem exist. We will construct a special oracle denoted as $O \diamond Q^{\text{Easy}}$ (explained later) such that:

1. The oracle $O \diamond Q^{\text{Easy}}$ is one-way. Thus, $G^{O \diamond Q^{\text{Easy}}}$ will be a PRG and $\Pi^{O \diamond Q^{\text{Easy}}}$ will be the HVZK system for the language $\mathcal{L} = \{Y : \exists X \text{ s.t. } Y = G^{O \diamond Q^{\text{Easy}}}(X)\}$.
2. There exist a $\ddot{Y} \notin \mathcal{L}$ (the false statement) and a $\mathbb{P}^{O \diamond Q^{\text{Easy}}}$ (the cheating prover \mathbb{P} with the oracle $O \diamond Q^{\text{Easy}}$) that is able to make $V^{O \diamond Q^{\text{Easy}}}(\ddot{Y})$ accept.

This will give us the desired contradiction as it breaks the soundness of the protocol $\Pi^{O \diamond Q^{\text{Easy}}}$.

Toward the above goal, we first sample two random oracles O, O' , a random string X , and compute $Y = G^O(X)$. Let $Q = \{(q_1, O(q_1)), \dots, (q_t, O(q_t))\}$ denote the query-answer pairs exchanged between G and its oracle O during computation $Y = G^O(X)$. We now define the oracle $O' \diamond Q(q) := \begin{cases} O(q) & \text{if } (q, O(q)) \in Q \\ O'(q) & \text{otherwise} \end{cases}$. It is not hard to verify that $Y = G^{O' \diamond Q}(X)$. By completeness, V will accept with probability $1 - \delta_c$ (where δ_c is the completeness error) in the execution $\text{Exec}_{X,Y}^{O' \diamond Q} = \langle P^{O' \diamond Q}(X, Y), V^{O' \diamond Q}(Y) \rangle$.

Note that during $\text{Exec}_{X,Y}^{O' \diamond Q}$, the verifier may make queries to its oracle $O' \diamond Q$. We define a set of “easy” queries:

$$Q^{\text{Easy}} := \{(q, O(q)) \mid V \text{ queries } q \text{ with “high” probability during } \text{Exec}_{X,Y}^{O' \diamond Q}\}.$$

Let Q^{Hard} be the set difference $Q \setminus Q^{\text{Easy}}$. It is not hard to see that $Y = G^{O' \diamond (Q^{\text{Easy}} \cup Q^{\text{Hard}})}(X)$. By completeness, V will accept with probability $1 - \delta_c$ in the execution $\text{Exec}_{X,Y}^{O' \diamond (Q^{\text{Easy}} \cup Q^{\text{Hard}})}$.

Now, consider the execution $\langle P^{O' \diamond (Q^{\text{Easy}} \cup Q^{\text{Hard}})}(X, Y), V^{O' \diamond Q^{\text{Easy}}}(Y) \rangle$, which is identical to $\text{Exec}_{X,Y}^{O' \diamond (Q^{\text{Easy}} \cup Q^{\text{Hard}})}$ except that we remove the Q^{Hard} from the verifier’s oracle. In this execution, the probability that V accepts will not differ too much from that in $\text{Exec}_{X,Y}^{O' \diamond (Q^{\text{Easy}} \cup Q^{\text{Hard}})}$, because the queries in Q^{Hard} are asked by V with only “low” probability.

We then prove that Y is in the range of $G^{O' \diamond Q^{\text{Easy}}}(\cdot)$ with probability at most 0.5 (up to negligible error). But the previous argument says that $V^{O' \diamond Q^{\text{Easy}}}(Y)$ accepts with probability close to 1. It then follows from an averaging argument that there exists “bad” \ddot{O}, \ddot{O}' and \ddot{X} ⁵ such that $\ddot{Y} = G^{\ddot{O}}(\ddot{X})$ is *not* in the range of $G^{\ddot{O}' \diamond \ddot{Q}^{\text{Easy}}}(\cdot)$, but $V^{\ddot{O}' \diamond \ddot{Q}^{\text{Easy}}}(\ddot{Y})$ can be convinced with probability close to 1, by the malicious prover $P^{\ddot{O}' \diamond (\ddot{Q}^{\text{Easy}} \cup \ddot{Q}^{\text{Hard}})}(\ddot{X}, \ddot{Y})$ (which can be viewed as an oracle machine $\mathbb{P}^{\ddot{O}' \diamond \ddot{Q}^{\text{Easy}}}$ with non-uniform advice \ddot{X}, \ddot{Y} , and \ddot{Q}^{Hard}). This breaks the soundness of $\Pi^{\ddot{O}' \diamond \ddot{Q}^{\text{Easy}}}$, thus completing the proof.

We remark that proving Y is in the range of $G^{O' \diamond Q^{\text{Easy}}}(\cdot)$ with probability ≤ 0.5 (up to negligible error) is the most involved part. And this is where the HVZK property of $\Pi^{(\cdot)}$ plays an essential role. Roughly, we will show that if this claim does not hold, then there exists an adversary $\mathcal{A}_{\text{PRG}}^O$ that can break the

⁵ Note that these values already determine the sets $\ddot{Q}, \ddot{Q}^{\text{Easy}}$, and \ddot{Q}^{Hard} as defined above.

pseudo-randomness of $G^{\mathcal{O}}(\cdot)$ by making *polynomially* many oracle queries. As we will explain later, this reduction requires $\mathcal{A}_{\text{PRG}}^{\mathcal{O}}$ to know the set Q^{Easy} w.r.t. the challenge string Y in the security game of PRG. But note that $\mathcal{A}_{\text{PRG}}^{\mathcal{O}}$ does not know the preimage X (if Y is indeed in the range), which is necessary to figure out Q^{Easy} . This is where the HVZK simulator comes to our rescue. We will run the simulator $\text{Sim}_V^{\mathcal{O}}(Y)$ repeatedly for (polynomially) many times to get an estimate \tilde{Q}^{Easy} for the set Q^{Easy} . This \tilde{Q}^{Easy} will be good enough to finish our proof. A more detailed overview of this strategy is provided in Sect. 4.4.

4.3 Proof of Thm. 2

Assume for contradiction that there exists an oracle machine $G^{(\cdot)}$ and a protocol $\langle P^{(\cdot)}, V^{(\cdot)} \rangle$ such that given the access to any one-way function $\{f_n\}_{n \in \mathbb{N}}$:

1. $G^{f_n} : \{0, 1\}^\ell \rightarrow \{0, 1\}^{\ell+1}$ is a PRG (ℓ and n are polynomially related); **and**
2. $\Pi = \langle P^{f_n}, V^{f_n} \rangle$ is a semi-honest zero-knowledge argument system for the Meta-FBB functionality $\mathcal{F}_{\text{ZK}}^{f_n}[G]$.

We first recall the following lemma, which says that the measure-one of randomly-sampled oracles is one-way.

Lemma 2 (One-Wayness of Random Oracles [25, 44]). *Let $\mathcal{O} = \{\mathcal{O}_n\}_{n \in \mathbb{N}}$ be a collection of oracles where each \mathcal{O}_n is chosen uniformly from the space of functions from $\{0, 1\}^n$ to $\{0, 1\}^n$. With probability 1 over the choice of \mathcal{O} , \mathcal{O} is one-way against unbounded adversaries that make only polynomially many oracle queries to \mathcal{O} .*

Let both $\mathcal{O} = \{\mathcal{O}_n\}_{n \in \mathbb{N}}$ and $\mathcal{O}' = \{\mathcal{O}'_n\}_{n \in \mathbb{N}}$ be defined (independently) as in Lem. 2. It follows from Lem. 2 that, with probability 1, both \mathcal{O} and \mathcal{O}' is one-way.

In the following, we show two hybrids. From the second hybrid, we will construct a malicious prover breaking the soundness of $\Pi^{(\cdot)}$ (with the oracle being instantiated by a special one-way oracle defined later). This will give us the desired contradiction, and thus will finish the proof of Thm. 2.

Notations. We first define some notations. For an oracle H and a set of tuples $S = \{(q_1, a_1), \dots, (q_t, a_t)\}$, we define a new oracle $H \diamond S$ as follows: if q equals some q_i for which there exists a pair (q_i, a_i) in the set S , the oracle $H \diamond S$ returns a_i ; otherwise, it returns $H(q)$. Formally,

$$H \diamond S(q) = \begin{cases} H(q) & \text{if } q \notin \{q_1, \dots, q_t\} \\ a_i & \text{if } q = q_i \in \{q_1, \dots, q_t\} \end{cases}$$

Hybrid H_0 . This hybrid samples $X_n \leftarrow \{0, 1\}^{\ell(n)}$, and computes $Y_n = G^{\mathcal{O}_n}(X_n)$. W.l.o.g., we assume that G on input X_n makes $t(n)$ *distinct* queries to its oracle \mathcal{O}_n , where $t(n)$ is a polynomial of n . Let $Q_n = \{(q_1, \mathcal{O}_n(q_1)), \dots, (q_t, \mathcal{O}_n(q_t))\}$ be the query-answer pairs during the computation $Y_n = G^{\mathcal{O}_n}(X_n)$.

Let $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond Q_n} = \langle P^{\mathcal{O}'_n \diamond Q_n}(X_n, Y_n), V^{\mathcal{O}'_n \diamond Q_n}(Y_n) \rangle$ denote the execution where P proves to V that there exists an X_n such that $Y_n = G^{\mathcal{O}'_n \diamond Q_n}(X_n)$. Note

that during this execution, the verifier may query to its oracle $O'_n \diamond Q_n$. For each $q_i \in \{0, 1\}^n$, let p_i denote the probability that q_i is queried by V during $\text{Exec}_{X_n, Y_n}^{O'_n \diamond Q_n}$. Let Q_n^{Easy} defines the set of “easy” queries and their corresponding answers:

$$Q_n^{\text{Easy}} := \left\{ (q_i, O'_n \diamond Q_n(q_i)) \mid p_i \geq \frac{1}{t(n) \cdot n} \text{ during } \text{Exec}_{X_n, Y_n}^{O'_n \diamond Q_n} \right\}. \quad (1)$$

Let Q_n^{Hard} be the set difference $Q_n \setminus Q_n^{\text{Easy}}$. We remark that Q_n and $Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}}$ may not be the same, but it must hold that $Q_n \subseteq Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}}$.

Looking ahead, we will instantiate $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ with the oracle $O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})$. Note that $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ will have the desired property only if they are instantiated with *one-way* functions. Therefore, we show in Claim 3 that the composed oracle $O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})$ is one-way. It is worth noting that the one-wayness of this composed oracle is independent of the choice of $\{X_n\}$, though the definition of Q_n , Q_n^{Easy} and Q_n^{Hard} depends on X_n .

Claim 3. *The collection of oracles $\{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})\}_{n \in \mathbb{N}}$ defined above is one-way with probability 1, where the probability is taken over the sampling of $\mathcal{O} = \{O_n\}_n$ and $\mathcal{O}' = \{O'_n\}_n$, and is independent of the distribution of $\{X_n\}_{n \in \mathbb{N}}$.*

Proof. The query-answer pairs in Q_n^{Easy} and Q_n^{Hard} are of the form $(q, O_n(q))$ or $(q, O'_n(q))$. Although X_n decides which $(q, *)$ ⁶ will be in Q_n^{Easy} and Q_n^{Hard} , the answer part $O_n(q)$'s and $O'_n(q)$'s are uniformly distributed, *independent* of X_n . That is, if O_n and O'_n are sampled randomly, then for any $X_n \in \{0, 1\}^{\ell(n)}$, $O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})$ will also be a random oracle. Therefore, for *any* $\{X_n\}_{n \in \mathbb{N}}$ where $X_n \in \{0, 1\}^{\ell(n)}$, the following holds

$$\{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})\}_{n \in \mathbb{N}} \stackrel{\text{i.d.}}{=} \{O''_n\}_{n \in \mathbb{N}},$$

where each O''_n is sampled uniformly from the space of functions from $\{0, 1\}^n$ to $\{0, 1\}^n$. Since it follows from Lem. 2 that $\{O''_n\}_{n \in \mathbb{N}}$ is one-way with probability 1, so is $\{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})\}_{n \in \mathbb{N}}$. \square

Claim 3 (together with our assumption) implies that, with probability 1 taken over the sampling of \mathcal{O} and \mathcal{O}' :

- $G^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})} : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell(n)+1}$ is pseudo-random against all (unbounded) adversaries that make polynomially many queries to the oracle $O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})$; **and**
- $\Pi^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$ is a semi-honest zero-knowledge argument system for the Meta-FBB functionality $\mathcal{F}_{\text{zk}}^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}[G]$.

Let $\text{Exec}_{X_n, Y_n}^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$ denote the execution $\langle P^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}(X_n, Y_n), V^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}(Y_n) \rangle$. Let $\text{Exec}_{X_n, Y_n}^{O'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})} = 1$ denote the event that the

⁶ The symbol “*” denotes the wildcard that matches any answer to q .

verifier accepts at the end of this execution. Claim 3 and the completeness of $\Pi_{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$ imply that:

$$\Pr_{\mathcal{O}, \mathcal{O}'} \left[\text{For sufficient large } n \in \mathbb{N}, \forall X_n \in \{0, 1\}^{\ell(n)}, Y_n = G^{\mathcal{O}_n}(X_n), \right] = 1, \quad (2)$$

$$\Pr_{\mathcal{O}, \mathcal{O}'} \left[\Pr \left[\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})} = 1 \right] \geq 1 - \delta_c(n) \right] = 1,$$

where the inner probability is taken over the random coins of the prover and the verifier during $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$, and $\delta_c(n)$ is the completeness error.

Hybrid H_1 . This hybrid is identical to the previous one, except that H_1 executes the protocol

$$\langle P^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}(X_n, Y_n), V^{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}}(Y_n) \rangle. \quad (3)$$

(Compared with the execution $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$ in H_0 , the only difference is that H_1 remove Q_n^{Hard} from the *verifier's* oracle.)

As mentioned in the **Proof Sketch** of Thm. 2, we want to show that the verifier accepts in Execution 3 with probability close to that in the execution $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$. This is formalized as Claim 4.

Claim 4. *With probability 1 taken over the sampling of \mathcal{O} and \mathcal{O}' , for sufficiently large $n \in \mathbb{N}$, it holds that $\forall X_n \in \{0, 1\}^{\ell(n)}$ and $Y_n = G^{\mathcal{O}_n}(X_n)$,*

$$\Pr \left[\langle P^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}(X_n, Y_n), V^{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}}(Y_n) \rangle = 1 \right] \geq \Pr \left[\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})} = 1 \right] - \frac{1}{n}, \quad (4)$$

where the probabilities in the above inequality are taken over the random coins of the prover and the verifier during the corresponding executions.

Proof. First, we remark that the “with probability 1” part in this claim is to ensure that $\{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})\}_n$ is one-way (see Claim 3). In the following, we proceed with $\{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})\}_n$ being one-way (so the probabilities below are not taken over \mathcal{O} and \mathcal{O}').

By definition, any query⁷ $q \in Q_n^{\text{Hard}}$ is asked by V during $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$ with probability $< \frac{1}{t(n) \cdot n}$. Let us denote the following event:

Event_{NoHard}: No $q \in Q_n^{\text{Hard}}$ is asked by V in $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$

It follows from union bound that

$$\Pr [\text{Event}_{\text{NoHard}}] \geq 1 - \frac{1}{n}, \quad (5)$$

where the probability is taken over the random coins of P and V in the execution $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$.

⁷ Technically, elements in Q_n^{Hard} are query-answer pairs. From here on, we override the notation “ \in ” such that $q \in Q_n^{\text{Hard}}$ means that there exists a pair $(q, *)$ in Q_n^{Hard} .

Now, we prove Inequality (4). In the following, for succinctness, let

- Exec_0 denote the execution $\langle P^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}(X_n, Y_n), V^{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}}(Y_n) \rangle$;
- Exec_1 denote the execution $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$.

Then, we have (probabilities below are taken over the random coins over P and V in the corresponding executions):

$$\begin{aligned} \Pr[\text{Exec}_1 = 1] &\geq \Pr[\text{Exec}_1 = 1 \mid \text{Event}_{\text{NoHard}}] \cdot \Pr[\text{Event}_{\text{NoHard}}] \\ &= \Pr[\text{Exec}_0 = 1 \mid \text{Event}_{\text{NoHard}}] \cdot \Pr[\text{Event}_{\text{NoHard}}] \end{aligned} \quad (6)$$

$$\geq \Pr[\text{Exec}_0 = 1] - \Pr[\neg \text{Event}_{\text{NoHard}}] \quad (7)$$

$$\geq \Pr[\text{Exec}_0 = 1] - \frac{1}{n} \quad (8)$$

where Step 6 is due to the fact that Exec_1 and Exec_0 are identical assuming V does not make any query $q \in Q_n^{\text{Hard}}$, Step 7 follows from the basic probability inequality that $\Pr[A \mid B] \cdot \Pr[B] \geq \Pr[A] - \Pr[\neg B]$, and Step 8 follows from Inequality (5).

This finishes the proof of Claim 4. \square

Claim 4 indicates that the verifier in Execution 3 accepts with “good” probability: at least as large as the accepting probability of $\text{Exec}_{X_n, Y_n}^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}$ minus $1/n$. Thus, we will have the desired contradiction if the Y_n in Execution 3 is a false statement, i.e. Y_n is not in the range of $G^{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}}$ (i.e. $G^{(\cdot)}$ instantiated by the verifier’s oracle in Execution 3). This argument is formalized and proved in Claims 5 and 6, which will eventually finish the proof of Thm. 2.

Claim 5. *Let Q_n^{Easy} be defined as in Expression (1). For sufficiently large $n \in \mathbb{N}$, the following holds:*

$$\Pr_{\mathcal{O}', X_n} \left[G^{\mathcal{O}'_n}(X_n) \in G^{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}}(\{0, 1\}^{\ell(n)}) \right] \leq \frac{1}{2} + \text{negl}(n). \quad (9)$$

Note that the above probability is taken (additionally) over $X_n \leftarrow \{0, 1\}^{\ell(n)}$.

Claim 6. *If Claim 5 holds, then Thm. 2 holds.*

The proof of Claim 5 is quite involved. It constitutes the main technical challenge of the current proof (of Thm. 2). Thus, we will deal with it in Sect. 4.4. In the following, we show the proof of Claim 6.

Proof of Claim 6. It follows from Expression (2) and Claim 4 that

$$\Pr_{\mathcal{O}, \mathcal{O}'} \left[\text{For sufficient large } n \in \mathbb{N}, \forall X_n \in \{0, 1\}^{\ell(n)}, Y_n = G^{\mathcal{O}'_n}(X_n), \right. \\ \left. \Pr \left[\langle P^{\mathcal{O}'_n \diamond (Q_n^{\text{Easy}} \cup Q_n^{\text{Hard}})}(X_n, Y_n), V^{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}}(Y_n) \rangle = 1 \right] \geq 1 - \frac{1}{n} - \delta_c(n) \right] = 1. \quad (10)$$

Following the same argument as for Claim 3, we can prove the one-wayness of the oracle $\{\mathcal{O}'_n \diamond Q_n^{\text{Easy}}\}_n$ as follows. For each $(q, \mathcal{O}_n(q)) \in Q_n^{\text{Easy}}$, the $\mathcal{O}_n(q)$ is a randomly sampled string from $\{0, 1\}^n$. Therefore, no matter what X_n is,

$O'_n \diamond Q_n^{\text{Easy}}$ is always a randomly sampled oracle (though Q_n^{Easy} is determined by X_n). It then follows from Lem. 2 that:

$$\Pr_{\mathcal{O}, \mathcal{O}'} \left[\forall X_n \in \{0, 1\}^{\ell(n)}, \{O'_n \diamond Q_n^{\text{Easy}}\}_{n \in \mathbb{N}} \text{ is one-way} \right] = 1. \quad (11)$$

By an averaging argument over Expressions (9) to (11), it follows that there exists fixed sequences $\{\ddot{O}_n\}_{n \in \mathbb{N}}$, $\{\ddot{O}'_n\}_{n \in \mathbb{N}}$ and $\{\ddot{X}_n\}_{n \in \mathbb{N}}$ ⁸ such that for sufficiently large $n \in \mathbb{N}$,

- $\{\ddot{O}_n \diamond \ddot{Q}_n^{\text{Easy}}\}_n$ is one-way; thus, $G^{\ddot{O}_n \diamond \ddot{Q}_n^{\text{Easy}}}$ is a PRG and $\Pi^{\ddot{O}_n \diamond \ddot{Q}_n^{\text{Easy}}}$ is an HVZK protocol for the membership of $G^{\ddot{O}_n \diamond \ddot{Q}_n^{\text{Easy}}}$; **and**
- \ddot{Y}_n is not in the range of $G^{\ddot{O}_n \diamond \ddot{Q}_n^{\text{Easy}}}$; **and**
- $\Pr \left[\langle P^{\ddot{O}'_n \diamond (\ddot{Q}_n^{\text{Easy}} \cup \ddot{Q}_n^{\text{Hard}})}(\ddot{X}_n, \ddot{Y}_n), V^{\ddot{O}'_n \diamond \ddot{Q}_n^{\text{Easy}}}(\ddot{Y}_n) \rangle = 1 \right] \geq 1 - \frac{1}{n} - \delta_c(n)$, where the probability is taken over the random coins of P and V .

Note that we can treat $P^{\ddot{O}'_n \diamond (\ddot{Q}_n^{\text{Easy}} \cup \ddot{Q}_n^{\text{Hard}})}(\ddot{X}_n, \ddot{Y}_n)$ as an oracle machine $\mathbb{P}^{\ddot{O}'_n \diamond \ddot{Q}_n^{\text{Easy}}}$, which has $(\ddot{Q}_n^{\text{Easy}}, \ddot{X}_n, \ddot{Y}_n)$ as non-uniform advice and makes only polynomially many queries to its oracle $\ddot{O}'_n \diamond \ddot{Q}_n^{\text{Easy}}$.

Since the completeness error $\delta_c(\cdot)$ is negligible, the above means that $\mathbb{P}^{\ddot{O}'_n \diamond \ddot{Q}_n^{\text{Easy}}}$ (with its non-uniform advice) convinces the verifier with non-negligible probability on the following *false* statement:

$$\ddot{Y}_n \in G^{\ddot{O}'_n \diamond \ddot{Q}_n^{\text{Easy}}}(\{0, 1\}^{\ell(n)}).$$

This contradicts the soundness of $\Pi^{\ddot{O}'_n \diamond \ddot{Q}_n^{\text{Easy}}}$. □

4.4 The Proof Sketch for Claim 5

Due to space constraints, we will present the formal proof for Claim 5 in the full version [35]. In this part, we provide an overview of it.

We assume for contradiction that Claim 5 is false and try to break the pseudo-randomness of G^{O_n} . First, observe that if $Y_n = G^{O_n}(X_n)$ where $X_n \leftarrow \{0, 1\}^{\ell(n)}$, then our assumption implies that Y_n is in the range of $G^{O'_n \diamond Q_n^{\text{Easy}}}(\cdot)$ with probability noticeably larger than 1/2. Therefore, on an input Y_n , if we can efficiently test if $Y_n \in G^{O'_n \diamond Q_n^{\text{Easy}}}(\{0, 1\}^{\ell(n)})$, we should have some advantage in the PRG game for $G^{O_n}(\cdot)$. This strategy has the following potential problems:

1. Without the preimage X_n , we *cannot* compute the set Q_n^{Easy} (see Expression (1)) using only polynomially many queries to O_n ;
2. If the input $Y_n \notin G^{O_n}(\{0, 1\}^{\ell(n)})$, the set Q_n (thus Q_n^{Easy}) is not even well-defined, as there is no preimage X_n .

To avoid using X_n , we will run the HVZK simulator to obtain an estimate of the set Q_n^{Easy} in the following way. Recall that Q_n^{Easy} contains the “easy”

⁸ Note that these values also fix the corresponding $\{\ddot{Y}_n\}_{n \in \mathbb{N}}$, $\{\ddot{Q}_n^{\text{Easy}}\}_{n \in \mathbb{N}}$ and $\{\ddot{Q}_n^{\text{Hard}}\}_{n \in \mathbb{N}}$ as in the above.

queries made by the verifier during $\text{Exec}_{X_n, Y_n}^{O'_n \diamond Q_n}$. By the HVZK property of the protocol $\Pi^{O'_n \diamond Q_n}$, each query in Q_n^{Easy} should be made with similar probability in the simulated execution $\text{Sim}_V^{O'_n \diamond Q_n}(Y_n)$. Therefore, repeating $\text{Sim}_V^{O'_n \diamond Q_n}(Y_n)$ (polynomially) many times will give us a good estimate to Q_n^{Easy} .

However, without X_n , we cannot figure out the set Q_n , which is necessary if we want to run $\text{Sim}_V^{O'_n \diamond Q_n}(Y_n)$. Fortunately, by a similar argument as that for Claim 3, we can prove that the oracle $\{O_n\}_n$ and $\{O'_n \diamond Q_n\}_n$ are identically distributed, even given X_n and $Y_n = G^{O_n}(X_n)$. Therefore, running $\text{Sim}_V^{O_n}(Y_n)$ will be just as good as running $\text{Sim}_V^{O'_n \diamond Q_n}(Y_n)$. Note that this also solves Problem 2, because the simulator still works when invoked on false statements.

Now, we can construct the PRG distinguisher $\mathcal{A}_{\text{PRG}}^{O_n}(Y_n)$ as follows: on input Y_n , $\mathcal{A}_{\text{PRG}}^{O_n}(Y_n)$ obtains an estimate $\tilde{Q}_n^{\text{Easy}}$ to Q_n^{Easy} by running $\text{Sim}_V^{O_n}(Y_n)$ for polynomially many times. It then samples a random function $O'_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$, and outputs 1 if $Y_n \in G^{O'_n \diamond \tilde{Q}_n^{\text{Easy}}}(\{0, 1\}^{\ell(n)})$; otherwise, it outputs 0. Note that although sampling O' requires exponential time, $\mathcal{A}_{\text{PRG}}^{O_n}(Y_n)$ only makes *polynomially many* queries to the oracle O_n .

If $Y_n = G^{O_n}(X_n)$ where $X_n \leftarrow \{0, 1\}^{\ell(n)}$, then by our assumption $\mathcal{A}^{O_n}(Y_n)$ outputs 1 with probability noticeably larger than $1/2$; if $Y_n \leftarrow \{0, 1\}^{\ell(n)+1}$, then Y_n is independent of O_n . Moreover, using a similar argument as for Claim 3, we can prove that Y_n is independent of the oracle $\tilde{Q}_n^{\text{Easy}}$ (thus $O'_n \diamond \tilde{Q}_n^{\text{Easy}}$). Since the function $G^{O'_n \diamond \tilde{Q}_n^{\text{Easy}}}(\cdot)$ stretch by 1 bit, the random Y_n will be in its range with probability $1/2$. This means $\mathcal{A}^{O_n}(Y_n)$ outputs 1 with probability exactly $1/2$.

This gives us the desired contradiction.

5 Proof-Based One-Way Functions

5.1 Definition

Definition 4 (Proof-Based OWFs). Let $\lambda \in \mathbb{N}$ be the security parameter. Let $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ be polynomials. A proof-based one-way function consists of a function $F_\lambda : \{0, 1\}^{a(\lambda)} \times \{0, 1\}^{b(\lambda)} \rightarrow \{0, 1\}^{c(\lambda)}$ and a protocol $\Pi = (S, R)$ of a pair of PPT machines. We use $(X, Y) \leftarrow \langle S(1^\lambda, x), R(1^\lambda, r) \rangle$ to denote the execution of protocol Π where the security parameter is λ , the inputs to S and R are x and r respectively, and the outputs of S and R are X and Y respectively. Let $Y = \perp$ denote that R aborts in the execution. The following conditions hold:

- **One-Wayness.** The function $\{F_\lambda\}_\lambda$ is one-way in the following sense:
 - Easy to compute: for all $\lambda \in \mathbb{N}$ and all $(x, r) \in \{0, 1\}^{a(\lambda)} \times \{0, 1\}^{b(\lambda)}$, $F_\lambda(x||r)$ can be computed in polynomial time on λ .
 - Hard to invert: for any non-uniform PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\cdot)$ such that $\forall r \in \{0, 1\}^{b(\lambda)}$,

$$\Pr \left[x \leftarrow \{0, 1\}^{a(\lambda)}, X^* \leftarrow \mathcal{A}(1^\lambda, F_\lambda(x||r)) : F_\lambda(x||r) = F_\lambda(X^*) \right] \leq \text{negl}(\lambda),$$

- **Completeness.** The protocol Π computes the ideal functionality \mathcal{F}_F defined in Fig. 1. Namely, $\forall \lambda \in \mathbb{N}$, $\forall x \in \{0, 1\}^{a(\lambda)}$ and $\forall r \in \{0, 1\}^{b(\lambda)}$, if $(X, Y) \leftarrow \langle S(1^\lambda, x), R(1^\lambda, r) \rangle$, then $X = x||r$ and $Y = F_\lambda(x||r)$.

The ideal functionality \mathcal{F}_F interacts with a sender S and a receiver R . Upon receiving the input $x \in \{0, 1\}^{a(\lambda)}$ from S and $r \in \{0, 1\}^{b(\lambda)}$ from R , the functionality \mathcal{F}_F sends $x||r$ to S , and $F(x||r)$ to R .

Fig. 1. Functionality \mathcal{F}_F for proof-based OWFs

- **Soundness.** For every PPT machine S^* and every auxiliary input $z \in \{0, 1\}^*$, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr \left[r \leftarrow \{0, 1\}^{b(\lambda)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, r) \rangle : Y \neq \perp \text{ and } \nexists x \text{ s.t. } F_\lambda(x||r) = Y \right] \leq \text{negl}(\lambda),$$

- **Zero-Knowledge.** This property is defined by requiring only security against corrupted R in the ideal-real paradigm for 2PC w.r.t. the ideal functionality \mathcal{F}_F in Fig. 1. Concretely, denote by $\text{REAL}_{\Pi, \mathcal{A}(z)}(1^\lambda, x, r)$ the random variable consisting of the output of S and the output of the adversary \mathcal{A} controlling R in an execution of Π , where x is the input to S and r to R . Similarly, denote by $\text{IDEAL}_{\mathcal{F}_F, \text{Sim}(z)}(1^\lambda, x, r)$ the corresponding output of S and Sim from the ideal execution. Then there exist a PPT simulator Sim such that for any PPT adversary \mathcal{A} , $\forall x \in \{0, 1\}^{a(\lambda)}$, $\forall r \in \{0, 1\}^{b(\lambda)}$, and $\forall z \in \{0, 1\}^*$, $\{\text{REAL}_{\Pi, \mathcal{A}(z)}(1^\lambda, x, r)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{\text{IDEAL}_{\mathcal{F}_F, \text{Sim}(z)}(1^\lambda, x, r)\}_{\lambda \in \mathbb{N}}$.

If the constructions of both F and Π makes only black-box access to other primitives, we call this a black-box PB-OWF.

5.2 Our Construction

Following the high-level idea described in Sect. 2.2, we show that PB-OWFs can be built assuming black-box access to OWFs.

Theorem 7 (Black-Box PB-OWFs from OWFs). *There exists a PB-OWF that satisfies Def. 4 and makes only black-box use of OWFs.*

Our construction consists of a one-way function F^f (Constr. 1) together with a protocol Π_F^f (Prot. 1). The construction relies on the following building blocks:

- a one-way function f ;
 - a zero-knowledge commit-and-prove protocol $\Pi_{\text{ZKCnP}} = (\text{BBCom}, \text{BBProve})$.
- Such protocols can also be constructed assuming only black-box access to f .

Remark 1 (On the Parameters in Constr. 1). The choice of $t(\lambda) = \log^2(\lambda)$ is somewhat arbitrary. In fact, any $t(\lambda) = \omega(\log \lambda)$ works as long as $(n - k - t)$ is some positive polynomial of λ for sufficiently large λ . This is to ensure that we can prove one-wayness and $(1 - \delta)^t$ is negligible on λ , which is needed when we prove soundness. We also remark that the role of r is to specify a size- t subset of $[n]$. The canonical way of mapping r to a size- t subset of $[n]$ may consume slightly less randomness than $|r| = t \log(n)$. For simplicity, we forgo further discussion and assume that there is a deterministic bijection between $\{0, 1\}^{t \log(n)}$ and all size- t subsets of $[n]$. Similarly, the $\{p_1, \dots, p_k\}$ are interpreted as a size- k subset of $[n]$, though we assign each p_i a length of $\log(n)$.

Construction 1: One-Way Function F^f

Let $m(\lambda)$ and $n(\lambda)$ be polynomials on λ . Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$ (see Rmk. 1). Assume that $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{m(\lambda)}$ is a one-way function. On input $x \in \{0, 1\}^{n\lambda + (\log(n)+m)k}$ and $r \in \{0, 1\}^{t \log(n)}$, F^f parses them as

$$x = (x_1, \dots, x_n) \parallel (p_1, y'_{p_1}), \dots, (p_k, y'_{p_k}), \text{ and } r = (b_1, \dots, b_t),$$

where $|x_i| = \lambda$, $|y'_{p_i}| = m$, $\{p_i\}_{i \in [k]}$ is a size- k subset of $[n]$, and $\{b_i\}_{i \in [t]}$ is a size- t subset of $[n]$. F^f computes via its oracle access to $f(\cdot)$ the values (y_1, \dots, y_n) , where $y_i = f(x_i)$ for all $i \in [n]$. Then, it computes $s = (s_1, \dots, s_n)$ as follows:

1. if $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} \neq \emptyset$, then let $s_i := y_i$ for all $i \in [n]$.
2. if $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} = \emptyset$, then let $s_i := \begin{cases} y'_i & i \in \{p_1, \dots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \dots, p_k\} \end{cases}$.

It finally outputs $Y = (s_1, \dots, s_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel (b_1, \dots, b_t)$.

Proof of Security. Due to space constraints, the complete proof for that (F^f, Π^f) satisfies Def. 4 will appear in the full version [35].

Note that Sect. 2.2 already contains the high-level idea for this proof. The one-wayness, completeness and ZK property follow from rather standard techniques. In the following, let us provide more details about the soundness proof.

First, note that the $r = \{b_1, \dots, b_t\}$ sent by R in Stage 3 is a size- t random subset of $[n]$. It will overlap with $\{p_1, \dots, p_k\}$ with negligible probability. Therefore, the **Editing** condition will almost never be triggered during a real execution of Prot. 1, thus can be safely ignored.

Stages 2 to 5 can be thought as the following cut-and-choose procedure: the sender computes $\{y_i = f(x_i)\}_{i \in [n]}$; then the receiver checks t of them randomly. This ensures that a malicious S^* cannot cheat on more than $k = \delta n$ of the y_i 's. We prove this statement formally in the full version [35], which requires us to handle extra technicalities due to the commit-and-prove structure and **Editing** condition. But this claim implies that a non-aborting Y output by an honest receiver contains at most $k = \delta n$ many s_i 's that does *not* have a preimage under f (except for negligible probability). Let us assume w.l.o.g. that there are exactly k such “no-preimage” s_i 's, which can be denoted as $\{s_{p_1}, \dots, s_{p_k}\}$ (i.e. we denote the indices of these no-preimage s_i 's by $\{p_1, \dots, p_k\}$). Then, for each s_i where $i \in [n] \setminus \{p_1, \dots, p_k\}$, this s_i must have (at least) one preimage under $f(\cdot)$. We denote an arbitrary preimage of such s_i as $f^{-1}(s_i)$. In particular, if i is equal to some $b_j \in \{b_1, \dots, b_t\}$, the Y already contains the preimage for s_{b_j} , which is x_{b_j} .

We emphasize that, conditioned on $Y \neq \perp$, we have $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} = \emptyset$. To see this, recall that R checks at Stage 5 that $y_{b_i} = f(x_{b_i})$ and $s_{b_i} = y_{b_i}$ for all $b_i \in \{b_1, \dots, b_t\}$. If there is a p_i falling in the set $\{b_1, \dots, b_t\}$, then $s_{p_i} (= y_{p_i})$ does not have a preimage under $f(\cdot)$. Then, R will output $Y = \perp$ at Stage 5.

With these observations, we show in the following how to construct x and r such that $F^f(x \parallel r) = Y$. At a high-level, we take advantage of Case 2. We will use the no-preimage s_{p_i} 's together with their indices as the (p_i, y'_{p_i}) part in x . We will set r to the $\{b_1, \dots, b_t\}$ contained in Y . Since $\{b_1, \dots, b_t\} \cap \{p_1, \dots, p_k\} = \emptyset$,

Protocol 1: Protocol Π_F^f for Our Proof-Based One-Way Function

Let f , m , n , t , and k be as in Constr. 1.

Input: the security parameter 1^λ is the common input. Sender S takes $x \in \{0, 1\}^{n\lambda + (\log(n)+m)k}$ as its private input; receiver R takes $r \in \{0, 1\}^{t \cdot \log(n)}$ as its private input.

1. S parses the input as $x = (x_1, \dots, x_n) \parallel (p_1, y'_{p_1}), \dots, (p_k, y'_{p_k})$, where $|x_i| = \lambda$ for all $i \in [n]$, $|y'_{p_j}| = m$ for all $j \in [k]$, and $\{p_i\}_{i \in [k]}$ forms a size- k subset of $[n]$. S defines a $2 \times n$ matrix $M = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$, where $y_i = f(x_i)$ for all $i \in [n]$.
2. S and R execute $\text{BBCom}(\alpha)$, the **Commit** stage of Π_{ZKCNP} , where S commits to the value

$$\alpha := M \parallel (p_1, y'_{p_1}), \dots, (p_k, y'_{p_k}). \quad (12)$$
3. R sends r to S .
4. S interprets r as a size- t subset $(b_1, \dots, b_t) \subseteq [n]$. S then defines $M_r = \begin{bmatrix} x_{b_1} & \cdots & x_{b_t} \\ y_{b_1} & \cdots & y_{b_t} \end{bmatrix}$, i.e. the columns of M specified by r . S also computes $\mathbf{s} = (s_1, \dots, s_n)$ in the way specified in Constr. 1. S sends to R the values M_r and \mathbf{s} .
5. With M_r , R checks (via its oracle access to $f(\cdot)$) if $f(x_{b_i}) = y_{b_i}$ holds for all $i \in [t]$; R also checks if $s_{b_i} = y_{b_i}$ holds for all $i \in [t]$. If all the checks pass, R proceeds to next step; otherwise, R halts and outputs \perp .
6. S and R execute BBProve , the **Prove** stage of Π_{ZKCNP} , where S proves that it performs Stage 4 honestly. Namely, S proves that the α committed at Stage 2 satisfies the following conditions:
 - (a) the values $\{p_1, \dots, p_k\}$ contained in α form a size- k subset of $[n]$; **and**
 - (b) the M_r does consist of the columns in M specified by r ; **and**
 - (c) The $\mathbf{s} = (s_1, \dots, s_n)$ satisfies the following conditions:
 - if $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} \neq \emptyset$, then $s_i = y_i$ for all $i \in [n]$.
 - if $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} = \emptyset$, then $s_i = \begin{cases} y'_i & i \in \{p_1, \dots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \dots, p_k\} \end{cases}$.

We remark that these conditions can indeed be expressed a predicate ϕ on the α committed at Stage 2. For completeness, we show the formal definition of ϕ in Fig. 2. It is also worth noting that predicate ϕ needs to have the values r and \mathbf{s} hard-wired, which are defined at Stages 3 and 4 respectively. This is why we need a Π_{ZKCNP} that allows us to defer the definition of the predicate until the Prove Stage.

7. (**Receiver's Output**). R outputs $Y = (s_1, \dots, s_n) \parallel (x_{b_1}, \dots, x_{b_t}) \parallel \{b_i\}_{i \in [t]}$.
8. (**Sender's Output**). S outputs $X = (x_1, \dots, x_n) \parallel (p_1, y'_{p_1}), \dots, (p_k, y'_{p_k}) \parallel \{b_i\}_{i \in [t]}$.

the function F^f will put the no-preimage s_{p_i} 's at the positions specified by p_i 's (according to Case 2), which will give us Y . Concretely, we set:

$$x = (x'_1, \dots, x'_n) \parallel (p_1, s_{p_1}), \dots, (p_k, s_{p_k}) \text{ and } r = (b_1, \dots, b_t),$$

$$\text{where } x'_i \text{'s are defined as follows: } \forall i \in [n], x'_i = \begin{cases} x_i & i \in \{b_1, \dots, b_t\} \\ 0^\lambda & i \in \{p_1, \dots, p_k\} \\ f^{-1}(s_i) & \text{otherwise} \end{cases}.$$

Predicate ϕ has the values $(\lambda, m, t, n, k, r, \mathbf{s})$ (defined in [Prot. 1](#)) hard-wired. On the input α , $\phi_{\lambda, m, t, n, k, r, \mathbf{s}}(\alpha) = 1$ if and only if all of the following hold:

- the α can be parsed as $M \parallel (p_1, y'_{p_1}), \dots, (p_k, y'_{p_k})$, where $M = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix}$ such that $|x_j| = \lambda$ and $|y_j| = m \forall j \in [n]$, $|p_i| = \log(n)$ and $|y'_{p_i}| = m \forall i \in [k]$; **and**
- the values $\{p_1, \dots, p_k\}$ form a size- k subset of $[n]$; **and**
- the M_r consists of the columns in M specified by r ; **and**
- the $\mathbf{s} = (s_1, \dots, s_n)$ satisfy the following requirement (recall that the $\{b_1, \dots, b_t\}$ are from r):
 - if $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} \neq \emptyset$, then $s_i = y_i$ for all $i \in [n]$.
 - if $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} = \emptyset$, then $s_i = \begin{cases} y'_i & i \in \{p_1, \dots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \dots, p_k\} \end{cases}$.

Fig. 2. Predicate $\phi_{\lambda, m, t, n, k, r, \mathbf{s}}(\cdot)$

We remark that $f^{-1}(s_i)$ may not be efficiently computable (indeed, f is a one-way function). But the above proof only relies on the *existence* of $f^{-1}(s_i)$. Also, we have $\{p_1, \dots, p_k\} \cap \{b_1, \dots, b_t\} = \emptyset$. It then follows from the description in [Constr. 1](#) (in particular, [Case 2](#)) that $F^f(x \parallel r) = Y$.

5.3 Proof-Based Pseudo-random Generators

We can also define proof-based pseudo-random generators (PB-PRGs) in a similar way as for PB-OWFs. It consists of a two-input function $G^g(\cdot, \cdot)$ and a protocol $\Pi_G^g = (S^g, R^g)$ such that for any PRG g , $G^g(\cdot, r)$ is a PRG for any choice of r , and Π_G^g satisfies the same completeness, soundness and ZK requirements as in [Def. 4](#) but w.r.t. G^g .

Our PB-PRG can be constructed by simply replacing the oracle OWF f with a PRG g in both [Constr. 1](#) and [Prot. 1](#) (our PB-OWF construction). There is one caveat: the output Y of [Constr. 1](#) contains the preimage x_{b_i} for y_{b_i} (or s_{b_i}). While this is fine for one-wayness, such a Y will not be pseudo-random, because an adversary can always learn if Y is in the range of $G^g(\cdot, r)$ by testing whether $y_{b_i} = g(x_{b_i})$. To fix this, in the output Y , we will place x_{b_i} in the position where we originally put y_{b_i} (and we can drop the $(x_{b_1}, \dots, x_{b_t})$ part from Y). We will show that this modification lead to a valid PB-PRG.

We present the definition, construction, and the security proof for PB-PRGs in the full version [\[35\]](#).

6 Proof-Based Collision-Resistant Hash Families

We now discuss proof-based collision-resistant hash families (PB-CRHF). As mentioned in Sect. 2.3, the definition and construction of PB-CRHF follow the same template as our PB-OWFs, except that we need to handle the **Editing** condition differently. Due to space constraints, we only show an overview of our PB-CRHF here. See the full version [35] for the details.

Construction 2: Collision-Resistant Hash Family $H_z^{h_i}$

Let $m(\lambda)$ and $n(\lambda)$ be polynomials of λ . Assume w.l.o.g. that n is a power of 2 (i.e., $n = 2^\ell$ for some ℓ). Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$ (see also Rmk. 1). Let $\mathcal{H}' = \{h_i\}_{i \in I}$ be a collision-resistant hash family where $h_i : \{0, 1\}^{2m(\lambda)} \rightarrow \{0, 1\}^{m(\lambda)}$. Denote its key generation as KGen' .

- **Key Generation.** On input 1^λ , sample a function from \mathcal{H}' by running $h_i \leftarrow \text{KGen}'(1^\lambda)$; sample a random string $z \leftarrow \{0, 1\}^{m(\lambda)}$; outputs (i, z) as the hash key.
- **Evaluation.** On input $\mathbf{x} \in \{0, 1\}^{nm+k(\log(2n)+m)+3m}$ and $\mathbf{r} \in \{0, 1\}^{t \log(n)}$, the evaluation algorithm parses them as:

$$\mathbf{x} = (x_1, \dots, x_n) \parallel (p_1, v_{p_1}), \dots, (p_k, v_{p_k}) \parallel \tau \parallel \mu, \quad \mathbf{r} = (b_1, \dots, b_t), \quad (13)$$

where $|x_i| = |v_{p_i}| = m$, $\{p_i\}_{i \in [k]}$ is a size- k subset of $[2n - 1]$, and $\{b_i\}_{i \in [t]}$ is a size- t subset of $[n]$. The set $\{b_i\}_{i \in [t]}$ specifies t leaves out of all the n leaves.

The algorithm builds a perfect binary tree T that has n leaves, where all the nodes are dummies. Note that the indices of the nodes in T are well-defined, even though T now contains only dummy nodes. The evaluation procedure outputs $Y = \mathbf{t}_1 \parallel \mathbf{t}_2 \parallel (\mathbf{P}_{b_1}, \dots, \mathbf{P}_{b_t}) \parallel (b_1, \dots, b_t)$, which is computed as follows:

1. **Non-Editing:** If $\tau = z$ or $h_i(\tau) \neq h_i(z)$ or $\text{Ind}(b_1, \dots, b_t) \cap \{p_1, \dots, p_k\} \neq \emptyset$:
 - (a) It fills the tree T as follows. It places (x_1, \dots, x_n) at the n leaves. For any other node in T , its content is the hash value under h_i on the concatenation of its left child and right child. Denote the root value as \mathbf{t}_1 .
 - (b) For $i \in [t]$, \mathbf{P}_{b_i} is the sibling path of leaf x_{b_i} in the above tree T ;
 - (c) Use h_i to hash⁹ the following A value to a length- m string denoted as \mathbf{t}_2 :

$$A = (p_1, v_{p_1}), \dots, (p_k, v_{p_k}) \parallel \tau \parallel \mu.$$

2. **Editing:** if $\tau \neq z$ and $h_i(\tau) = h_i(z)$ and $\text{Ind}(b_1, \dots, b_t) \cap \{p_1, \dots, p_k\} = \emptyset$:
 - (a) It fills the tree T as follows. It places (x_1, \dots, x_n) at the n leaf positions in T . Then, fill the tree bottom up, following the rule for Merkle tree (i.e. the hashing of two children nodes' contents is the parent node's content), with the following exception: for node $p_i \in \{p_1, \dots, p_k\}$, it fills node p_i with the v_{p_i} contained in \mathbf{x} , instead of the hash of the children of node p_i . Denote the root value as \mathbf{t}_1 .
 - (b) For $i \in [t]$, \mathbf{P}_{b_i} is defined as the sibling path of leaf x_{b_i} in the tree T ;
 - (c) Set $\mathbf{t}_2 = \mu$ (recall that μ is contained in \mathbf{x});

⁹ Note that the input to h_i should have length $2m$. But $|A| > 2m$. This can be handled using domain-extension techniques, e.g., the Merkle-Damgård transformation [5, 37].

Protocol 2: Protocol $\Pi_z^{h_i}$ for Our PB-CRHF

Let \mathcal{H}' , m , n , δ , t and k be as in Constr. 2. Let $\Pi_{\text{ZK}_{\text{CNP}}} = (\text{BBCom}, \text{BBProve})$ be a black-box commit-and-prove protocol. For a function defined by (i, z) from the PB-CRHF in Constr. 2, this protocol proceeds as follows. Both parties take the security parameter 1^λ as the common input. Sender S takes a string $x \in \{0, 1\}^{nm+k(\log(n)+m)+3m}$ as private input; receiver R takes a string $r \in \{0, 1\}^{t \log(n)}$ as private input.

1. S parses x as $(x_1, \dots, x_n) \parallel (p_1, v_{p_1}), \dots, (p_k, v_{p_k}) \parallel \tau \parallel \mu$ (in the same manner as in Expression (13)). S build a Merkle tree $MT'_{h,m}(x)$ using (x_1, \dots, x_n) as the leaves (this is identical to Step 1a). Denote the root of this tree as \mathbf{t}_x .
2. S and R execute $\text{BBCom}(\nu)$, the **Commit** stage of $\Pi_{\text{ZK}_{\text{CNP}}}$, where S commits to the following value

$$\nu := \mathbf{t}_x \parallel (p_1, \dots, p_k). \quad (14)$$

3. R sends the value r .
4. S parses r as (b_1, \dots, b_t) where each b_i is of length $\log(n)$. With the values x and r , S evaluates the function $H_z^{h_i}$ as per Constr. 2 to compute the following Y , which it sends to R :

$$Y = \mathbf{t}_1 \parallel \mathbf{t}_2 \parallel (\mathbf{P}_{b_1}, \dots, \mathbf{P}_{b_t}) \parallel (b_1, \dots, b_t).$$

5. R checks if \mathbf{P}_{b_i} is Merkle-consistent for all $i \in [t]$. R aborts if any of the check fails.
6. S and R execute BBProve , the **Prove** stage of $\Pi_{\text{ZK}_{\text{CNP}}}$, where S proves that the ν committed in Stage 2 satisfies the following conditions:
 - (a) the $\{p_1, \dots, p_k\}$ in ν form a size- k subset of $[2n-1]$, where $k = \delta n$; **and**
 - (b) the \mathbf{t}_x contained in τ is equal to \mathbf{t}_1 , **or** $\text{Ind}(b_1, \dots, b_t) \cap \{p_1, \dots, p_k\} = \emptyset$.
7. (**Receiver's Output**). R outputs $Y = \mathbf{t}_1 \parallel \mathbf{t}_2 \parallel (\mathbf{P}_{b_1}, \dots, \mathbf{P}_{b_t}) \parallel (b_1, \dots, b_t)$.
8. (**Sender's Output**). S outputs

$$X = (x_1, \dots, x_n) \parallel (p_1, v_{p_1}), \dots, (p_k, v_{p_k}) \parallel \tau \parallel \mu \parallel (b_1, \dots, b_t).$$

On the Definition. Our PB-CRHF consists of an oracle machine $H^{(\cdot)}$ and an oracle protocol $\Pi_H^{(\cdot)}$. As mentioned in Sect. 2.3, the $H^{(\cdot)}$ will be instantiated as a hash *family*. That is, given a collision-resistant hash family \mathcal{H}' , we first run its KGen' to sample a function $h_i \in \mathcal{H}'$, and then instantiate $H^{(\cdot)}$'s oracle as h_i . Therefore, H^{h_i} is also a hash family whose KGen simple runs the KGen' for \mathcal{H}' (and samples a random string z that we will discuss later).

Once $H^{(\cdot)}$ and $\Pi_H^{(\cdot)}$ are instantiated with an $h_i \leftarrow \text{KGen}'(1^\lambda)$, we can start talking about the security. Same as in Def. 4, H^{h_i} takes two inputs x and r . We require that, for all r , $H^{h_i}(\cdot, r)$ is collision-resistant on its first input. The protocol $\Pi_H^{h_i}$ satisfies the similar completeness, soundness and ZK requirement as in Def. 4. We provide a formal definition in the full version [35].

Our Construction. The formal construction is provided in Constr. 2 and Prot. 2. We follow the high-level idea described in Sect. 2.3 with the following modifications. Instead of hashing the (x_1, \dots, x_n) (contained in x) separately, we build a Merkle tree using them as the leaves. In Constr. 2, \mathbf{P}_i denotes the sibling path from leaf x_i to the root; $\text{Ind}(b_1, \dots, b_t)$ denotes the set of *indices*

of the nodes on path P_{b_1}, \dots, P_{b_t} (see the full version [35] for more details). In Prot. 2, the receiver checks t leaves and their corresponding sibling paths. This ensures that there are at least $(n - k)$ “good” leaves, in the sense that there are valid sibling paths from the Merkle root to them. In the **Editing** case, this will allow us to perform preimage editing by planting the v_{p_i} values on the k “bad” paths to obtain a (partial) tree consistent with the root t_1 contained in Y . Note that we also hash the Λ in Step 1c. As explained in Sect. 2.3, this is to prevent the adversary from taking advantage of preimage editing to find collisions.

It is also worth noting that Constr. 2 and Prot. 2 work for an x of fixed length. But since we hash the $\{x_i\}_{i \in [n]}$ part using a Merkle tree, we can handle x with a various-length $\{x_i\}_{i \in [n]}$ part (which dominates the length of x). To maintain security, we simply include in Y the height of the Merkle tree.

Proof of Security. The security can be proved in a similar manner as for our PB-OWFs. We provide the formal security proof in the full version [35].

References

1. Barak, B., Mahmoody-Ghidary, M.: Merkle’s key agreement protocol is optimal: an $O(n^2)$ attack on any key agreement from random oracles. *J. Cryptol.* **30**(3), 699–734 (2017). <https://doi.org/10.1007/s00145-016-9233-9>
2. Chatterjee, R., Liang, X., Pandey, O.: Improved black-box constructions of composable secure computation. In: Czumaj, A., Dawar, A., Merelli, E. (eds.) *ICALP 2020. LIPIcs*, vol. 168, pp. 28:1–28:20. Schloss Dagstuhl, July 2020. <https://doi.org/10.4230/LIPIcs.ICALP.2020.28>
3. Cook, S.A.: The complexity of theorem-proving procedures. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151–158 (1971)
4. Crépeau, C., Kilian, J.: Achieving oblivious transfer using weakened security assumptions (extended abstract). In: *29th FOCS*, pp. 42–52. IEEE Computer Society Press, October 1988. <https://doi.org/10.1109/SFCS.1988.21920>
5. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) *CRYPTO 1989. LNCS*, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39
6. Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) *CRYPTO 2005. LNCS*, vol. 3621, pp. 378–394. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_23
7. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: *22nd ACM STOC*, pp. 416–426. ACM Press (1990). <https://doi.org/10.1145/100216.100272>
8. Garg, S., Gupta, D., Miao, P., Pandey, O.: Secure multiparty RAM computation in constant rounds. In: Hirt, M., Smith, A. (eds.) *TCC 2016-B, Part I. LNCS*, vol. 9985, pp. 491–520. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53641-4_19
9. Garg, S., Kiyoshima, S., Pandey, O.: A new approach to black-box concurrent secure computation. In: Nielsen, J.B., Rijmen, V. (eds.) *EUROCRYPT 2018, Part II. LNCS*, vol. 10821, pp. 566–599. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_19

10. Garg, S., Liang, X., Pandey, O., Visconti, I.: Black-box constructions of bounded-concurrent secure computation. In: Galdi, C., Kolesnikov, V. (eds.) SCN 2020. LNCS, vol. 12238, pp. 87–107. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57990-6_5
11. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_15
12. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge (2001)
13. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
14. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC, pp. 25–32. ACM Press (1989). <https://doi.org/10.1145/73007.73010>
15. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th FOCS, pp. 174–187. IEEE Computer Society Press, October 1986. <https://doi.org/10.1109/SFCS.1986.47>
16. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC, pp. 291–304. ACM Press, May 1985. <https://doi.org/10.1145/22145.22178>
17. Goyal, V.: Constant round non-malleable protocols using one way functions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC, pp. 695–704. ACM Press, June 2011. <https://doi.org/10.1145/1993636.1993729>
18. Goyal, V., Lee, C.K., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: a black-box approach. In: 53rd FOCS, pp. 51–60. IEEE Computer Society Press, October 2012. <https://doi.org/10.1109/FOCS.2012.47>
19. Goyal, V., Ostrovsky, R., Scafuro, A., Visconti, I.: Black-box non-black-box zero knowledge. In: Shmoys, D.B. (ed.) 46th ACM STOC, pp. 515–524. ACM Press, May/June 2014. <https://doi.org/10.1145/2591796.2591879>
20. Haitner, I.: Semi-honest to malicious oblivious transfer—the black-box way. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_23
21. Hästad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999)
22. Hazay, C., Venkatasubramanian, M.: On the power of secure two-party computation. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 397–429. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_14
23. Hazay, C., Venkatasubramanian, M.: Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 263–285. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03807-6_10
24. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In: 21st ACM STOC, pp. 12–24. ACM Press, May 1989. <https://doi.org/10.1145/73007.73009>
25. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, pp. 44–61. ACM Press, May 1989. <https://doi.org/10.1145/73007.73012>

26. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Kleinberg, J.M. (ed.) 38th ACM STOC, pp. 99–108. ACM Press, May 2006. <https://doi.org/10.1145/1132516.1132531>
27. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC, pp. 21–30. ACM Press, June 2007. <https://doi.org/10.1145/1250790.1250794>
28. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32
29. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
30. Khurana, D., Ostrovsky, R., Srinivasan, A.: Round optimal black-box “Commit-and-Prove”. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 286–313. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03807-6_11
31. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC, pp. 20–31. ACM Press, May 1988. <https://doi.org/10.1145/62212.62215>
32. Kiyoshima, S.: Round-efficient black-box construction of composable multi-party computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 351–368. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_20
33. Kiyoshima, S.: Round-optimal black-box commit-and-prove with succinct communication. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 533–561. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_19
34. Levin, L.A.: Universal sequential search problems. *Problemy Peredachi Informatsii* **9**(3), 115–116 (1973)
35. Liang, X., Pandey, O.: Towards a unified approach to black-box constructions of zero-knowledge proofs. *Cryptology ePrint Archive*, Report 2021/836 (2021). <https://eprint.iacr.org/2021/836>
36. Lin, H., Pass, R.: Black-box constructions of composable protocols without set-up. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 461–478. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_27
37. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
38. Naor, M.: Bit commitment using pseudo-randomness. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 128–136. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_13
39. Pass, R., Wee, H.: Black-box constructions of two-party protocols from one-way functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_24
40. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_1
41. Rosulek, M.: Must you know the code of f to securely compute f ? In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 87–104. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_7

42. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: 51st FOCS, pp. 531–540. IEEE Computer Society Press, October 2010. <https://doi.org/10.1109/FOCS.2010.87>
43. Wyner, A.D.: The wire-tap channel. *Bell Syst. Tech. J.* **54**(8), 1355–1387 (1975)
44. Yerukhimovich, A.B.: A study of separations in cryptography: new results and new models. Ph.D. thesis, University of Maryland, College Park, Maryland, USA (2011)



Compressing Proofs of k -Out-Of- n Partial Knowledge

Thomas Attema^{1,2,3(✉)}, Ronald Cramer^{1,2(✉)}, and Serge Fehr^{1,2(✉)}

¹ CWI, Cryptology Group, Amsterdam, The Netherlands
`{cramer, serge.fehr}@cwi.nl`

² Leiden University, Mathematical Institute, Leiden, The Netherlands

³ TNO, Cyber Security and Robustness, The Hague, The Netherlands
`thomas.attema@tno.nl`

Abstract. In a proof of partial knowledge, introduced by Cramer, Damgård and Schoenmakers (CRYPTO 1994), a prover knowing witnesses for some k -subset of n given public statements can convince the verifier of this claim without revealing which k -subset. Their solution combines Σ -protocol theory and linear secret sharing, and achieves linear communication complexity for general k, n . Especially the “one-out-of- n ” case $k = 1$ has seen myriad applications during the last decades, e.g., in electronic voting, ring signatures, and confidential transaction systems.

In this paper we focus on the discrete logarithm (DL) setting, where the prover claims knowledge of DLs of k -out-of- n given elements. Groth and Kohlweiss (EUROCRYPT 2015) have shown how to solve the special case $k = 1$ with *logarithmic* (in n) communication, instead of linear as prior work. However, their method takes explicit advantage of $k = 1$ and does not generalize to $k > 1$. Alternatively, an *indirect* approach for solving the considered problem is by translating the k -out-of- n relation into a circuit and then applying communication-efficient circuit ZK. For $k = 1$ this approach has been highly optimized, e.g., in ZCash.

Our main contribution is a new, simple honest-verifier zero-knowledge proof protocol for proving knowledge of k out of n DLs with *logarithmic* communication and *for general k and n* , without requiring any generic circuit ZK machinery. Our solution puts forward a novel extension of the *compressed* Σ -protocol theory (CRYPTO 2020), which we then utilize to compress a new Σ -protocol for proving knowledge of k -out-of- n DL’s down to logarithmic size. The latter Σ -protocol is inspired by the CRYPTO 1994 approach, but a careful re-design of the original protocol is necessary for the compression technique to apply. Interestingly, *even for $k = 1$ and general n* our approach improves prior *direct* approaches as it reduces prover complexity without increasing the communication complexity. Besides the conceptual simplicity, we also identify regimes of practical relevance where our approach achieves asymptotic and concrete improvements, e.g., in proof size and prover complexity, over the generic approach based on circuit-ZK.

Finally, we show various extensions and generalizations of our core result. For instance, we extend our protocol to proofs of partial knowledge of Pedersen (vector) commitment openings, and/or to include a proof that the witness satisfies some additional constraint, and we show how to extend our results to non-threshold access structures.

1 Introduction

1.1 Proofs of Partial Knowledge

Proofs of partial knowledge [14] allow a prover to convince a verifier that the prover knows k out of n secrets, without revealing which secrets the prover knows. Typically, these secrets are solutions to public instances of intractable problems, such as the discrete logarithm problem. The work of [14] gives an elegant solution with linear communication complexity that combines Σ -protocol theory with linear secret sharing. Our goal is to invoke the techniques of Bulletproofs [6, 9] and follow-up work, in particular the compressed Σ -protocol framework of [2], to construct proofs of partial knowledge with *logarithmic* communication complexity.

Compressed Σ -protocol theory [2] was introduced as a strengthening of Σ -protocol theory. It inherits the flexibility and versatility of Σ -protocols while compressing their communication complexity from linear to logarithmic. The main pivot of this theory is a standard Σ -protocol for opening linear forms on Pedersen vector commitments, i.e., a Σ -protocol for proving that a committed vector \mathbf{x} satisfies $L(\mathbf{x}) = y$ for a public linear form L and a public scalar y . By an appropriate adaptation of the techniques from [6, 9] this pivotal Σ -protocol is compressed to achieve communication complexity that is logarithmic in the dimension of \mathbf{x} ; additionally a linearization approach to handle non-linearities is described [2]. As one of the applications of this theory it was shown how to obtain circuit zero-knowledge (ZK) protocols with logarithmic communication complexity for arbitrary arithmetic circuits.

An obvious approach for constructing proofs of partial knowledge with logarithmic complexity is to apply recent advances in communication efficient circuit ZK to a suitably constructed circuit for capturing the k -out-of- n relation. For instance, this is how the decentralized and confidential transaction system ZCash is designed [22].

In this work here, we take a more *direct* approach that avoids generic circuit techniques. We find such a direct solution scientifically more appealing, but there are also efficiency considerations that may make our solution the preferred choice (we discuss this in detail in Sect. 1.5). Our solution is inspired by the core idea of [14] of exploiting properties of linear secret sharing; however, the straightforward approach of compressing the original [14]-protocol with the techniques of [2] does not work; the third message in the [14]-protocol includes a consistent secret sharing of the challenge, which cannot be compressed.

1.2 Their Applications

Proofs of partial knowledge have seen myriad applications during the last decades. For instance, they were shown to be applicable to the construction of *group signature schemes* [10]. Group signature schemes [11] allow a member of a group to sign a message without revealing which member it is, while a designated *group manager* is capable of revoking the anonymity of the signer.

Another application is to *ring signature schemes* [29], which do not contain such a revocation mechanism. In a ring signature scheme, a group member can select any ad-hoc subset of group members and anonymously sign a message on behalf of this subset. Here, 1-out-of- n proofs together with the Fiat-Shamir [16] heuristic allow for a straightforward construction of ring-signature schemes. Because of the ad-hoc nature a ring signature must contain a list of the subset’s members and, therefore, its size grows linearly in the size of the ring; however, in many practical scenarios the costs of specifying a ring can be amortized over many instances.

Proofs of partial knowledge, in particular in the form of ring signature schemes, also play a crucial role in confidential decentralized transactions system such as Zerocoin [28]. Zerocoin was proposed as an extension of Bitcoin to provide stronger privacy guarantees. A Zerocoin transaction requires a ZKPoK that the transferred coin is an element of a public set of unspent coins. Other decentralized payment systems that rely on 1-out-of- n proofs to provide confidentiality are, e.g., Lelantus [23], ZCash [22], Zether [8] and Monero [31].

As a generalization of ring signature schemes, threshold ring signatures only allow a large enough subset to compute a valid signature [7]. These schemes require a generalization of the special proof of partial knowledge case $k = 1$. For instance, Monero is actively working on threshold ring signature schemes [19]. Moreover, in [15], it is shown how their generalization from 1-out-of- n proofs to so-called many-out-of-many proofs improves the communication complexity of the Zether payment system. They show that many practical scenarios require more general proofs of partial knowledge than only 1-out-of- n proofs.

1.3 Our Contributions

In this work, we start off by introducing and analyzing a novel extension of the core compression protocol from compressed Σ -protocol theory [2]. Namely, we observe that the compressed Σ -protocol for opening *linear forms* can be adapted to apply to general *homomorphisms*, i.e., for proving that a committed vector $\mathbf{x} \in \mathbb{Z}_q^n$ satisfies $f(\mathbf{x}) = y$ for an arbitrary group homomorphism $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}$ and an element $y \in \mathbb{G}$. The loss of efficiency is at most a constant factor and the adapted protocol still achieves a logarithmic communication complexity. Furthermore, the amortization technique to open *multiple* linear forms for essentially the price of one [2] directly carries over to opening multiple homomorphisms. This generalized functionality has not been considered before in the context of logarithmic complexity. As we discuss below, it turns out to be very useful in the design of efficient proofs of partial knowledge, but possibly also beyond.

Indeed, given n group elements $P_1, \dots, P_n \in \mathbb{G}$, consider a prover claiming that it knows k out of the n DLs, i.e., it knows a subset $S \subset \{1, \dots, n\}$ of cardinality k and exponents $x_i \in \mathbb{Z}_q$ such that $g^{x_i} = P_i$ for all $i \in S$. Inspired by the design principle of [14], we reduce this k -out-of- n case to the n -out-of- n case by having the prover “eliminate” the instances that it does not know, and then we apply the amortized version of the new compressed Σ -protocol for opening homomorphisms to prove the n instances in one go, with logarithmic

complexity. However, the original solution of [14] to reduce the k -out-of- n to the n -out-of- n case, achieved by secret sharing the challenge, does not work for us, as the resulting protocol is not in the shape for the (above generalization of the) [2] compression technique to apply.

Instead, we use the following new solution. The prover first chooses an $(n-k+1)$ -out-of- n Shamir secret sharing (s_1, \dots, s_n) of the default secret $s = 1$, where it selects the non-constant “random” coefficients a_1, \dots, a_{n-k} of the sharing polynomial $p(X) = 1 + a_1X + \dots + a_{n-k}X^{n-k}$ so that $s_i = 0$ for $i \notin S$. The prover then commits to the vector $(\mathbf{a}, \mathbf{t}) = (a_1, \dots, a_{n-k}, t_1, \dots, t_n) \in \mathbb{Z}_q^{2n-k}$ with t_i set to $t_i = s_i x_i$ for any i , understood to be equal to 0 for $i \notin S$, i.e., when $s_i = 0$. Finally, it proves that

$$g^{t_i} = P_i^{s_i} \quad (1)$$

for all $i \in \{1, \dots, n\}$. Proving this linear relation with a standard Σ -protocol gives a novel secret-sharing based realization of [14], with linear communication complexity.

This protocol crucially differs from the original proofs-of-partial knowledge, making it amenable to our compression techniques. First, it generates a single compact commitment to the vector of interest (\mathbf{a}, \mathbf{t}) . No compact commitments are used in the original protocol. Second, the $(n-k+1, n)$ -secret sharing of 1 is *implicitly* defined by the committed coefficients a_1, \dots, a_{n-k} . By contrast, in the original protocol the prover computes an arbitrary $(n-k+1, n)$ -secret sharing of a challenge c sampled uniformly at random by the verifier. Since the verifier has to check the consistency of this secret sharing this approach has an inherent linear communication complexity.

Let us consider our proofs-of-partial knowledge realization. By the linearity of Shamir’s secret sharing scheme, for any i Eq. 1 can be cast as a homomorphism of the committed values $a_1, \dots, a_{n-k}, t_1, \dots, t_n$,¹ and thus our novel variation of [2], including the amortization over the n homomorphisms, applies, thereby achieving a logarithmic communication complexity.

In total, our k -out-of- n proof protocol requires the prover to send $4 \lceil \log_2(2n-k+1) \rceil - 5$ group elements and 4 elements in \mathbb{Z}_q to the verifier. We also show how to further reduce this to $2 \lceil \log_2(2n-k+1) \rceil - 1$ group elements and 4 elements in \mathbb{Z}_q on a pairing-based platform. The protocol is public-coin and can therefore be made non-interactive by the Fiat-Shamir transform [16]. The public set-up, necessary for the vector commitment, consists of at most $2n$ group elements, and the complexity of the prover scales linearly in n .

1.4 Extensions and Variations

The conceptual simplicity of our design principle makes it easy to extend the protocol in various directions, for instance to proofs of partial knowledge about

¹ Concretely, we consider the homomorphism $f_i : \mathbb{Z}_q^{2n-k} \rightarrow \mathbb{G}, (\mathbf{a}, \mathbf{t}) \mapsto g^{t_i} P_i^{-\ell_i(\mathbf{a})}$ with ℓ_i the linear functional $\ell_i(\mathbf{a}) = a_1 i + a_2 i^2 + \dots + a_{n-k} i^{n-k}$, and we ask the prover to prove that the committed values map to P_i .

“multi-generator discrete logarithms” and corresponding Pedersen vector commitments. Furthermore, by introducing a pairing and considering a pairing based extension of Pedersen’s vector commitment scheme, we can reduce the relevant constant by another factor up to 2. Moreover, we show that our proofs of partial knowledge are compatible with circuit ZK protocols of [2], allowing the prover to demonstrate that his secret information satisfies some arbitrary given constraint. Finally, we generalize the results from threshold access structures to arbitrary access structures.

1.5 Comparison with Other Approaches

Achieving partial proofs of knowledge with logarithmic complexity has received quite some attention over the last few years, with different approaches and different (partial) solutions. We discuss here the examples that are most relevant in the context of our new approach, and we compare them with our results.

In [21], Groth and Kohlweiss consider the special case $k = 1$, and where the prover claims to be able to open 1 out of n public commitments to zero. Their solution is a Σ -protocol that works for any additively homomorphic commitment scheme over \mathbb{Z}_q and it achieves a logarithmic communication complexity. To describe their approach, let $1 \leq \ell \leq n$ be the index of the prover’s secret. The prover commits to each bit of ℓ and runs $\lceil \log_2(n) \rceil$ standard Σ -protocols, in parallel and on a common challenge, proving that all these commitments can indeed be opened to a binary value. In addition, the prover shows that the responses of these parallel Σ -protocols satisfy some multiplicative relation, which completes the protocol. This approach does not have an obvious generalization to k -out-of- n proofs.

The 1-out-of- n proof of [21] requires the prover to send $4 \lceil \log_2(n) \rceil$ group elements and $3 \lceil \log_2(n) \rceil + 1$ field elements. By using Pedersen *vector* commitments, instead of ordinary Pedersen commitment, the communication costs can be further reduced to $\lceil \log_2(n) \rceil + 4$ group elements and $\lceil \log_2(n) \rceil + 3$ field elements [5]. Instead of the binary decomposition, the approach of [5] considers the m -ary decomposition of the secret index ℓ . Here, we have optimized their approach for proving knowledge of 1-out-of- n discrete logarithms by taking $m = 2$. The work of [5] focuses on a slightly different scenario in which the communication costs are minimized for $m = 4$.

In comparison, in our protocol, which works for any k , the prover sends $4 \lceil \log_2(2n - k + 1) \rceil - 5$ group elements and 4 field elements to the verifier, which is reduced to $2 \lceil \log_2(2n - k + 1) \rceil - 1$ group elements and 4 field elements on a pairing-based platform. Hence, perhaps surprisingly, our simple protocols are comparable to dedicated solutions for the special case $k = 1$.

Recently, a generalization from 1-out-of- n proofs of [21] to “many-out-of-many” proofs was given [15]. This generalization considers a prover that claims to know the opening of all commitments in one of the orbits of a public permutation of n public commitments. However, the protocol only works for permutations with orbits of equal size. Since the permutation is public and of this specific form, this protocol does not constitute a general k -out-of- n proof of partial knowledge.

The prover complexity of the aforementioned 1-out-of- n proofs [5, 21] is $O(n \log(n))$, and that of the “many-out-of-many” proofs [15] is $O(n \log^2(n))$. By contrast, our protocol has prover complexity $O(n)$. Here and below, we express the prover complexity in terms of the number of group exponentiations required, explaining why these complexities are independent of the security parameter κ . However, note that the size of the group and therefore the complexity of evaluating a single group exponentiation does depend on κ . Similarly, we specify the communication costs in terms of the number group and field elements.

Aiming to improve the prover complexity of the 1-out-of- n proofs of [5, 21], Jivanyan and Mamikonyan [24] proposed a hierarchical approach. Their protocol assumes that $n = NM$ and applies an appropriate 1-out-of- N proof followed by a 1-out-of- M proof. It reduces the prover complexity from $O(n \log(n))$ down to $O(n + N \log(N) + M \log(M))$, which equals $O(n)$ if, for example, $N = M = \sqrt{n}$. However, this hierarchical approach increases the communication complexity to $O(N \log(N) + M \log(M) + M)$, hence it is subject to a trade-off between prover and communication complexity.

Alternatively to our and the above approaches, proofs of partial knowledge can be constructed via generic circuit ZK protocols. This *indirect* approach is, for example, followed by the confidential transaction system ZCash [22]. A standard construction for the 1-out-of- n case is to incorporate the group elements P_i into a Merkle tree [27], and ask the prover to prove knowledge of an exponent x_i such that the group element g^{x_i} is the leaf of a valid, but secret, Merkle path. In this case, the arithmetic circuit C implements a composition of the exponentiation g^{x_i} and the $\log_2(n)$ hash function evaluations corresponding to the validation of a Merkle path, and it is therefore of size $|C| = O(\kappa \log(n))$, where κ is the security parameter.

Even though this is obviously possible, to our knowledge this Merkle-tree approach has not been explicitly generalized to the k -out-of- n case before, making it difficult to do a rigorous efficiency comparison. However, such a generalization would require k Merkle paths to be validated, resulting in circuits of size $|C| = O(\kappa k \log_2(n))$. In addition, the circuit has to validate that the k Merkle paths are *distinct*. If the (public) Merkle tree is constructed such that its leaves are in a sorted order, this requires a circuit of size $O(k)$. In these complexity estimates we neglect the $O(\kappa n)$ size circuit required to construct the Merkle tree, because these costs can be amortized in some applications.

In Table 1, the asymptotic complexities of our direct approach are compared with the indirect approach, instantiated with typical communication efficient circuit ZK protocols for which the size of the public parameters and the prover complexity are linear and the communication complexity is logarithmic in the circuit size. We observe that, if $k = \Omega(n/\log(n))$, our approach yields an asymptotic improvement over the indirect approach.

Moreover, the constants of our approach are small. By contrast, taking for instance the case $k = 1$ and a highly optimized group, associated to a security parameter $\kappa \approx 100$, the indirect approach can be instantiated with arithmetic circuits containing approximately $1400 \log_2(n)$ multiplication gates [22]. Hence, even for the 1-out-of- n case, where the indirect approach has better asymptotic

complexities, we obtain better communication complexity for n ranging up to roughly 9000.

Table 1. Comparison of the asymptotic complexities of the indirect approach, using typical communication-efficient circuit ZK protocols, and our direct approach, for k -out-of- n proofs of partial knowledge, with security parameter κ . The size of the public parameters and the communication complexity are expressed in the number of group and field elements. The prover complexity is expressed in the number of group exponentiations.

	Indirect Circuit ZK Approach	Our Direct Approach
Size of Public Parameters	$O(\kappa k \log n)$	$O(n)$
Prover Complexity	$O(\kappa k \log n)$	$O(n)$
Communication Complexity	$O(\log(\kappa k \log n))$	$O(\log n)$

The above circuit approach can further be adjusted, for instance by invoking ZK protocols with constant communication complexity [20], or by replacing Merkle trees with RSA-accumulators [4], which results in arithmetic circuits with a number of multiplication gates that is constant in n [30]. However, these approaches are incomparable in that they are based on computational hardness assumptions that are considered less standard, like the strong-RSA assumption or the knowledge-of-exponent assumption. Furthermore, in these protocols, the size of the public parameters and the prover complexity are still linear in the circuit size, and for practical instances still result in sizeable circuits, respectively.

1.6 Organization of the Paper

The remainder of paper is organized as follows. In Sect. 2, we recall the notation and some of the results from compressed Σ -protocol theory [2]. In Sect. 3, we describe our twist on the pivotal Σ -protocol from [2]. In Sect. 4, we combine this generalization with an adaptation of the techniques from [14] to construct our proof of partial knowledge. Finally, in Sect. 5, we discuss a number of extensions and generalizations of our proofs of partial knowledge.

2 Preliminaries

2.1 Interactive Proofs

We briefly introduce the concept of an interactive proof² and some of the basic (security) properties. An interactive proof Π for relation R is a protocol between prover \mathcal{P} and a verifier \mathcal{V} . It takes as public input the statement x and as prover’s private input the witness w , which is written as $\text{INPUT}(x; w)$. As the output of

² In contrast to the original definition [18], we do not require an interactive proof to be complete and sound by definition; instead, we consider those (and other) properties as desirable security properties.

the protocol the verifier either accepts or rejects the prover’s claim of knowing a witness w . Π is called (perfectly) *complete* if on any input $(x; w) \in R$ the verifier always accepts. Evaluating Π on input $(x; w)$ is also written as $\Pi(x; w)$.

An interactive proof with μ communication rounds is also called a μ -move protocol. Note that the final message is always sent from the prover to the verifier. The messages communicated in one protocol evaluation are also referred to as a *conversation* or a *transcript*. If all of the verifier’s random coins are made public, one speaks of a *public-coin* protocol. All our protocols will be public-coin and thereby suitable for the Fiat-Shamir transformation [16], which turns public-coin interactive proofs into *non-interactive* protocols.

An interactive proof Π for relation R is said to have *witness extended emulation* [26] if there exists algorithm χ (witness extended emulator) that runs in expected polynomial time and does the following. The algorithm χ , on input x and given rewindable oracle access to a (possibly dishonest) prover \mathcal{P}^* , outputs a transcript and a witness w such that: (1) the emulated transcript is statistically indistinguishable from conversations between \mathcal{P}^* and an honest verifier \mathcal{V} , and (2) the probability that the emulated transcript is accepting and the witness w is not a valid witness for x is negligible. Witness extended emulation gives a notion for *proofs of knowledge* (PoKs) that is sufficient in practical applications [2, 6, 9].

We also consider the computational version of a PoK, where witness extended emulation is required to hold only for computationally bounded dishonest provers under a computational hardness assumption. In those cases, the relation R typically depends on a (possibly implicit) security parameter, as well as on some additional public parameters that are assumed to be chosen according to a specific probability distribution, and the success probability of the prover is then understood to be on average over the choice of these public parameters. These computational variants of proofs of knowledge are also called arguments of knowledge.

Protocol Π is called *honest verifier zero-knowledge* (HVZK) if there exists an efficient simulator that, on input a statement x that admits a witness w , outputs an accepting transcript, such that the simulated transcripts follow exactly the same distribution as transcripts between an honest prover and an honest verifier.

A 3-move public-coin interactive proof is called a Σ -protocol. The 3 messages are then typically denoted (a, c, z) where c is called the *challenge*. For a HVZK Σ -protocol the simulator often proceeds by first selecting a random challenge c and then preparing the messages a and z ; in this case, we speak of *special honest verifier zero-knowledge* (SHVZK).

A Σ -protocol is called *k-special sound* if there exists an efficient algorithm that, on input any statement x and k accepting transcripts $(a, c_1, z_1), \dots, (a, c_k, z_k)$ with common first message a and pairwise distinct challenges c_i , outputs a witness w for x .

More generally, we consider $(2\mu + 1)$ -move public-coin protocols, in which all the verifier’s messages are uniformly random challenges. These protocols are called (k_1, \dots, k_μ) -*special sound* if there exists an efficient algorithm that, on input any statement x and a (k_1, k_2, \dots, k_μ) -tree of accepting transcripts, out-

puts a witness w for x . A (k_1, k_2, \dots, k_μ) -tree of accepting transcripts is a set of $\prod_{i=1}^\mu k_i$ accepting transcripts that are arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges correspond to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf node.

We note that in some public-coin protocols the verifier sends μ challenges in less than $2\mu + 1$ rounds, i.e., some of the verifier's messages contain more than one challenge. For these protocols, we also consider the (k_1, \dots, k_μ) -special soundness property. In this case, a (k_1, k_2, \dots, k_μ) -tree of accepting transcripts contains nodes that do not correspond to a message sent from the prover to the verifier.

Let us assume that the challenges are sampled uniformly at random from challenge sets with a cardinality that is exponential in the security parameter. In this work all challenge sets are equal to $\mathbb{Z}_q \cong \mathbb{Z}/(q\mathbb{Z})$ for some prime q that is understood to be exponential in the security parameter. Hence, for the protocols in this work this assumption is satisfied. Then witness extended emulation is known to follow from (k_1, k_2, \dots, k_μ) -special soundness [6].³ In this work, we will show that all protocols are (k_1, k_2, \dots, k_μ) -special sound for some μ and some list of k_i 's, from which witness extended emulation therefore follows.

2.2 Multi-exponentiation and the Pedersen Vector Commitment Scheme

We consider statements over the ring $\mathbb{Z}_q \cong \mathbb{Z}/(q\mathbb{Z})$ with q prime. We let \mathbb{G} be an Abelian group of prime order q for which we write its group operation multiplicatively. We write vectors in \mathbb{Z}_q^n or \mathbb{G}^n in boldface, i.e., $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{Z}_q^n$ and $\mathbf{g} := (g_1, \dots, g_n) \in \mathbb{G}^n$, and we write $\mathbf{g}^{\mathbf{x}}$ for the multi-exponentiation

$$\mathbf{g}^{\mathbf{x}} := \prod_{i=1}^n g_i^{x_i} \in \mathbb{G}.$$

Furthermore, for vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$ and scalar $c \in \mathbb{Z}_q$, we have the following component-wise operations:

$$\begin{aligned} \mathbf{g} * \mathbf{h} &:= (g_1 h_1, g_2 h_2, \dots, g_n h_n) \in \mathbb{G}^n, \\ \mathbf{g}^c &:= (g_1^c, g_2^c, \dots, g_n^c) \in \mathbb{G}^n, \\ \mathbf{x} * \mathbf{y} &:= (x_1 y_1, x_2 y_2, \dots, x_n y_n) \in \mathbb{Z}_q^n. \end{aligned}$$

Additionally, assuming n is even, we write $\mathbf{g}_L := (g_1, \dots, g_{n/2})$, $\mathbf{g}_R := (g_{n/2+1}, \dots, g_n) \in \mathbb{G}^{n/2}$ and $\mathbf{x}_L := (x_1, \dots, x_{n/2})$, $\mathbf{x}_R := (x_{n/2+1}, \dots, x_n) \in \mathbb{Z}_q^{n/2}$, for the left and right halves of these vectors.

³ In a recent unpublished work [3], it is shown that (k_1, k_2, \dots, k_μ) -special soundness implies the more standard notion of *knowledge soundness*. In turn, knowledge soundness implies witness extended emulation [26].

We let \mathbb{G}_T be another Abelian group and denote the set of all group homomorphisms $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T$ by $\text{Hom}(\mathbb{Z}_q^n, \mathbb{G}_T)$. Typically $\mathbb{G}_T = \mathbb{G}$ or $\mathbb{G}_T = \mathbb{Z}_q$, in the latter case $\text{Hom}(\mathbb{Z}_q^n, \mathbb{G}_T) = \text{Hom}(\mathbb{Z}_q^n, \mathbb{Z}_q)$ is the set of linear forms on \mathbb{Z}_q^n . For any homomorphism $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T$ it holds that its image $\text{im}(f) \subset \mathbb{G}_T$ is a \mathbb{Z}_q -module. For this reason, and without loss of generality, we assume that \mathbb{G}_T is a \mathbb{Z}_q -module.

Moreover, we define the left and right part of f as follows:

$$\begin{aligned} f_L : \mathbb{Z}_q^{n/2} &\rightarrow \mathbb{G}_T, & \mathbf{x} &\mapsto f(\mathbf{x}, 0), \\ f_R : \mathbb{Z}_q^{n/2} &\rightarrow \mathbb{G}_T, & \mathbf{x} &\mapsto f(0, \mathbf{x}), \end{aligned} \tag{2}$$

where, e.g., $(\mathbf{x}, 0) \in \mathbb{Z}_q^n$ is the vector $\mathbf{x} \in \mathbb{Z}_q^{n/2}$ appended with $n/2$ zeros.

In this work we also consider the Pedersen vector commitment scheme. This commitment scheme allows a prover to (compactly) commit to an n -dimensional vector $\mathbf{x} \in \mathbb{Z}_q^n$ in a single group element $P \in \mathbb{G}$. We recall that a Pedersen vector commitment P is simply a multi-exponentiation, i.e.,

$$P = h^\gamma \mathbf{g}^{\mathbf{x}},$$

for public parameters $h \in \mathbb{G}$ and $\mathbf{g} \in \mathbb{G}^n$ and for a (private) $\gamma \in \mathbb{Z}_q$ sampled uniformly at random by the prover.

The Pedersen vector commitment scheme is perfectly hiding and computationally binding under the discrete logarithm assumption. More precisely, the commitment scheme is binding under the assumption that a prover does not know a non-zero vector $(\gamma, x_1, \dots, x_n) \in \mathbb{Z}_q^{n+1}$ such that

$$h^\gamma \prod_{i=1}^n g_i^{x_i} = 1.$$

Such a non-zero vector $(\gamma, x_1, \dots, x_n)$ is also called a non-trivial discrete log relation for group elements h, g_1, \dots, g_n . From here on forward, we assume that these group elements have been sampled uniformly at random in a setup phase and that the prover does not know a non-trivial discrete logarithm (DL) relation. These group elements form the set of public parameters for all our protocols. We say a protocol is computationally (k_1, \dots, k_μ) -special sound, under the discrete logarithm assumption, if (k_1, \dots, k_μ) -special soundness holds under the assumption that a prover does not know a non-trivial DL relation between the public parameters.

3 Proving Group Homomorphism Openings on Multi-exponentiations

In this section, we construct an interactive proof for proving that a secret multi-exponent $\mathbf{x} \in \mathbb{Z}_q^n$ for a public multi-exponentiation $P = \mathbf{g}^{\mathbf{x}} \in \mathbb{G}$ is mapped to a given public value y under an arbitrary but given group homomorphism $f : \mathbb{Z}_q^n \rightarrow$

\mathbb{G}_T . Our new protocol has a communication complexity that is logarithmic in the dimension n . By considering one of the coordinates of \mathbf{x} to be “the randomness”, and considering an f that ignores this coordinate, we immediately get a protocol that applies to Pedersen vector commitments and proves that the committed vector satisfied the relation defined by the considered group homomorphism and the target value y .

Our approach for constructing said protocol is as follows. We start with the canonical Σ -protocol for the considered problem of proving $f(\mathbf{x}) = y$ (Sect. 3.1), and we then adapt the compression mechanism of [2] such that it is applicable to our setting. Indeed, our setting is a generalization of [2], which applies to the special case where f is a linear form $L : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$. This then results in a compressed Σ -protocol that features the claimed logarithmic complexity (Sect. 3.3).

Later in the section, we also discuss a couple of (standard) amortization techniques applied to our protocol, for instance for proving $f_i(\mathbf{x}) = y_i$ for *several* group homomorphisms f_i at (essentially) the cost of proving one.

3.1 The Standard Σ -protocol for Opening Homomorphisms

We consider the problem of proving that the multi-exponent \mathbf{x} of a multi-exponentiation $P = \mathbf{g}^{\mathbf{x}}$ is mapped to a certain value y under a given homomorphism $f \in \text{Hom}(\mathbb{Z}_q, \mathbb{G}_T)$, i.e., that $f(\mathbf{x}) = y$, without revealing \mathbf{x} . More concretely, we want to construct PoK protocols for the relation

$$R_f = \{ (P \in \mathbb{G}, y \in \mathbb{G}_T; \mathbf{x} \in \mathbb{Z}_q^n) : P = \mathbf{g}^{\mathbf{x}}, y = f(\mathbf{x}) \}. \quad (3)$$

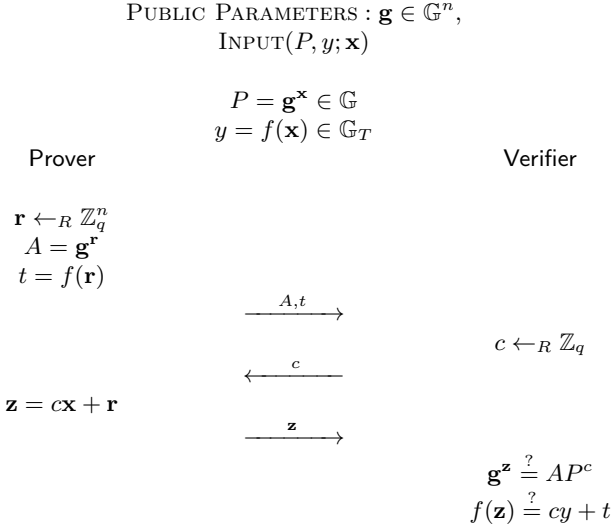
Protocol 1, denoted by Π_0 , is the canonical Σ -protocol for this relation R_f , following the generic construction design for q -one-way group homomorphisms⁴ [12,13]. The properties of Π_0 , known to hold for this generic construction, are summarized in Theorem 1. Note that the only difference between this protocol and Protocol 2 of [2] is that here we consider multi-exponentiations and general group homomorphisms instead of Pedersen commitments and linear forms.

Theorem 1 (Homomorphism Evaluation). *Π_0 is a Σ -protocol for relation R_f . It is perfectly complete, special honest-verifier zero-knowledge and unconditionally special sound. Moreover, the communication costs are:*

- $\mathcal{P} \rightarrow \mathcal{V}$: 1 element of \mathbb{G} , 1 element of \mathbb{G}_T and n elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: 1 element of \mathbb{Z}_q .

⁴ Here, applied to the q -one-way group homomorphisms $\mathbb{Z}_q^n \rightarrow \mathbb{G} \times \mathbb{G}_T$, $\mathbf{x} \mapsto (\mathbf{g}^{\mathbf{x}}, f(\mathbf{x}))$.

Protocol 1. Σ -protocol Π_0 for relation R_f
 Opening a homomorphism on a Pedersen vector commitment.



3.2 Compression Mechanism

The Σ -protocol Π_0 for opening homomorphisms has a linear communication complexity. We now deploy the techniques from [2, 6, 9] to compress the communication complexity from linear to logarithmic. A first observation is that the verifier’s final check verifies that

$$(AP^c, cy + t; \mathbf{z}) \in R_f,$$

i.e., that the prover’s final message \mathbf{z} is a witness with respect to the same relation R_f for the statement $(AP^c, cy + t)$; which is computed by the verifier. This is no coincidence; this holds generically for this standard construction of Σ -protocols for q -one-way group homomorphisms. The final message of Π_0 can therefore be understood as a trivial PoK for relation R_f , and replacing this trivial PoK by a more efficient one will reduce the communication complexity without affecting security (significantly). In particular, the alternative PoK does not have to be zero-knowledge since the trivial one obviously is not.

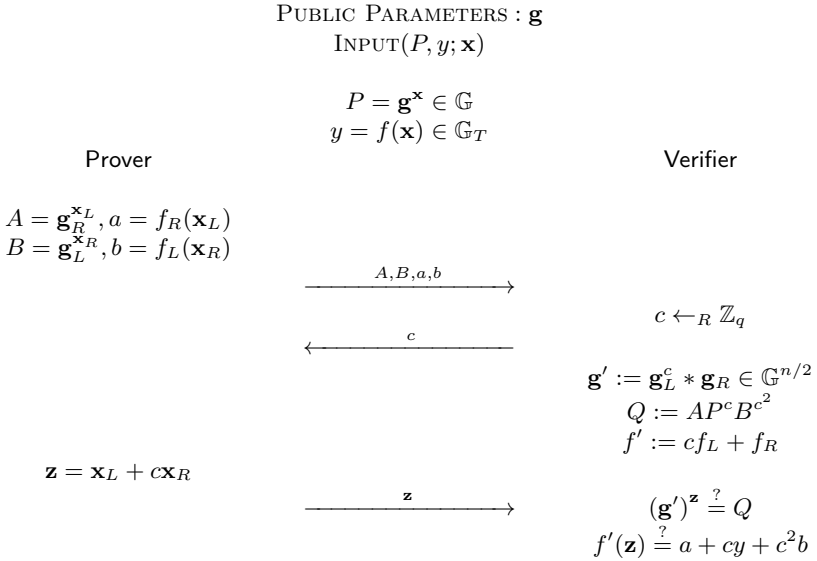
Our compression mechanism is thus an interactive proof Π_1 for relation R_f that is not zero-knowledge anymore but has improved efficiency. The compression mechanism is very similar to the one used in [2]. The difference is that we consider the more general case of opening arbitrary group homomorphisms, rather than restricting ourselves to linear forms. This generalization requires a minor adaptation. The first step in the compression of [2] is namely to incorporate the linear form evaluation into the multi-exponentiation as an additional exponent on a new generator $k \in \mathbb{G}$. This reduction step does not apply to

the general case of opening arbitrary group homomorphisms, and is therefore omitted in our protocols. For this reason we directly apply (a minor adaptation of) the main compression mechanism of [2]; ultimately this will increase the communication costs of the compressed Σ -protocol by roughly a factor two when compared to opening linear forms. However, in contrast to the compressed Σ -protocol for opening linear forms [2], our protocol is unconditionally sound rather than computationally. In Sect. 5.1, we show how a more general class of homomorphisms can be incorporated into the commitment, thereby avoiding the factor two loss in the communication efficiency.

The compression mechanism, i.e., our protocol Π_1 for relation R_f that has improved efficiency but is not zero-knowledge, is described in Protocol 2 below. Here, n is assumed to be even, which is without loss of generality (if not the witness can be appended with a zero). Also, recall that $\mathbf{x}_L := (x_1, \dots, x_{n/2})$ equals the left half of the vector $\mathbf{x} \in \mathbb{Z}_q^n$ and that $f_R(\mathbf{x}_L) := f(0, \dots, 0, \mathbf{x}_L)$, etc.

Before discussing the security of Π_1 as a proof of knowledge in Theorem 2, we emphasize the following two important properties of Π_1 . The size of the response has halved compared to the original protocol Π_0 , and thereby the communication costs are reduced by roughly a factor two, and second, verifying the correctness of the response is again by means of checking whether it is a witness for the relation $R_{f'}$, now instantiated with the group homomorphism $f' := cf_L + f_R \in \text{Hom}(\mathbb{Z}_q^{n/2}, \mathbb{G}_T)$.

Protocol 2. Compression Mechanism Π_1 for relation R_f .



Theorem 2 (Compression Mechanism). *Let $n \in \mathbb{Z}_{>0}$ be even. Then Π_1 is a 3-move protocol for relation R_f . It is perfectly complete and unconditionally 3-special sound. Moreover, the communication costs are:*

- $\mathcal{P} \rightarrow \mathcal{V}$: 2 elements of \mathbb{G} , 2 elements of \mathbb{G}_T and $n/2$ elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: 1 element of \mathbb{Z}_q .

Proof. **Completeness** follows directly.

Special Soundness: We show that the protocol is 3-special sound, i.e., there exists an efficient algorithm that on input three accepting transcripts computes a witness for relation R_f .

Let $(A, B, a, b, c_1, \mathbf{z}_1)$, $(A, B, a, b, c_2, \mathbf{z}_2)$ and $(A, B, a, b, c_3, \mathbf{z}_3)$ be three accepting transcripts for distinct challenges $c_1, c_2, c_3 \in \mathbb{Z}_q$. Let $a_1, a_2, a_3 \in \mathbb{Z}_q$ be such that

$$\begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^2 & c_2^2 & c_3^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Note that, since the challenges are distinct, this Vandermonde matrix is invertible and a solution to this equation exists. We define $\bar{\mathbf{z}} = \sum_{i=1}^3 a_i(c_i \mathbf{z}_i, \mathbf{z}_i)$ for which it is easily verified that

$$\mathbf{g}^{\bar{\mathbf{z}}} = P \quad \text{and} \quad f(\bar{\mathbf{z}}) = y.$$

Hence, $\bar{\mathbf{z}}$ is a witness for relation R_f , which completes the proof. \square

3.3 Compressed Σ -protocol

Finally, we compose Σ -protocol Π_0 and its compression mechanism Π_1 to obtain a compressed Σ -protocol for opening homomorphisms on multi-exponentiations $\mathbf{g}^{\mathbf{x}}$ such as Pedersen vector commitments. We follow the notation of [2] and write $\Pi_b \diamond \Pi_a$ for the composition of two composable interactive proofs Π_a and Π_b . Protocols Π_a and Π_b are composable if protocol Π_b is a PoK for the prover's final message of protocol Π_a . Recall that this composition means that the final message of protocol Π_a is replaced by an execution of protocol Π_b .

We assume that n is a power of two, if it is not the witness can be appended with zeros such that its dimension is a power of 2. For $n \leq 2$ it is optimal to omit the compression mechanism, for this reason it is assumed that $n > 2$. To minimize the communication complexity we recursively apply the compression protocol Π_1 until the dimension of the witness is reduced to four, i.e., $\mu = \lceil \log_2(n) \rceil - 2$ times. For this composition we write

$$\Pi_c = \underbrace{\Pi_1 \diamond \cdots \diamond \Pi_1}_{\mu \text{ times}} \diamond \Pi_0. \tag{4}$$

Theorem 3 captures the security and efficiency properties of Protocol Π_c .

Theorem 3 (Compressed Σ -Protocol for Opening Homomorphisms).

Let $n > 2$, then Π_c is a $(2\mu + 3)$ -move protocol for relation R_f , where $\mu = \lceil \log_2(n) \rceil - 2$. It is perfectly complete, special honest-verifier zero-knowledge and unconditionally $(2, 3, 3, \dots, 3)$ -special sound. Moreover, the communication costs are:

- $\mathcal{P} \rightarrow \mathcal{V}$: $2 \lceil \log_2(n) \rceil - 3$ elements of \mathbb{G} , $2 \lceil \log_2(n) \rceil - 3$ elements of \mathbb{G}_T and 4 elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: $\lceil \log_2(n) \rceil - 1$ elements of \mathbb{Z}_q .

Proof. **Completeness** follows in a straightforward manner.

Special Honest Verifier Zero-Knowledge follows since Π_0 is SHVZK. A simulator for Π_c runs the simulator for Π_0 , and replaces the final messages of the simulated transcripts by honest executions of $\Pi_1 \diamond \dots \diamond \Pi_1$.

Special Soundness: Since the protocol is the composition of protocols that are 2- or 3-special sound, it is easily seen that Π_c is $(2, 3, \dots, 3)$ -special sound, i.e., there exists an efficient algorithm that on input a $(2, 3, \dots, 3)$ -tree (depth $\mu + 1$) of $2 \cdot 3^\mu$ accepting transcripts computes a witness for relation R_f . \square

Remark 1. We explicitly emphasize once more that the above and below results on opening homomorphisms $f(\mathbf{x})$ on multi-exponentiations $\mathbf{g}^{\mathbf{x}}$ immediately carry over to opening homomorphisms $f(\mathbf{x})$ on Pedersen vector commitments $\mathbf{g}^{\mathbf{x}}h^\gamma$, simply by renaming the involved variables in the obvious way.

3.4 Amortization Techniques

This section describes two standard amortization techniques. First, we consider the scenario where a prover wishes to open *one* homomorphism f on *many* multi-exponentiations P_1, \dots, P_s , i.e., we consider the relation

$$R_{\text{AMOREXP}} = \left\{ (P_1, \dots, P_s, y_1, \dots, y_s; \mathbf{x}_1, \dots, \mathbf{x}_s) : \right. \\ \left. P_1 = \mathbf{g}^{\mathbf{x}_1}, y_1 = f(\mathbf{x}_1), \dots, P_s = \mathbf{g}^{\mathbf{x}_s}, y_s = f(\mathbf{x}_s) \right\}. \quad (5)$$

The standard (amortized) Σ -protocol for relation R_{AMOREXP} is similar to Σ -protocol Π_0 for relation R_f : it has the same first two moves, but then the prover's final response is $\mathbf{z} = \mathbf{r} + \sum_{i=1}^s c^i \mathbf{x}_i$ and the verifier checks that $\mathbf{g}^{\mathbf{z}} = AP_1^c P_2^{c^2} \dots P_s^{c^s}$ and $f(\mathbf{z}) = t + cy_1 + c^2 y_2 + \dots + c^s y_s$. The communication costs of the amortized Σ -protocol are exactly equal to the communication costs of protocol Π_0 and the compression mechanism applies as before. We denote the compressed amortized Σ -protocol for relation R_{AMOREXP} by Π_{AMOREXP} . Its main properties are summarized in Theorem 4.

Theorem 4 (Amortization over Many Multi-Exponentiations). Let $n > 2$, then Π_{AMOREXP} is a $(2\mu + 3)$ -move protocol for relation R_{AMOREXP} , where $\mu = \lceil \log_2(n) \rceil - 2$. It is perfectly complete, special honest-verifier zero-knowledge and unconditionally $(s + 1, 3, 3, \dots, 3)$ -special sound. Moreover, the communication costs are:

- $\mathcal{P} \rightarrow \mathcal{V}$: $2 \lceil \log_2(n) \rceil - 3$ elements of \mathbb{G} , $2 \lceil \log_2(n) \rceil - 3$ elements of \mathbb{G}_T and 4 elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: $\lceil \log_2(n) \rceil - 1$ elements of \mathbb{Z}_q .

Second, we consider the amortization scenario where a prover wishes to open *many* homomorphisms f_1, \dots, f_s on *one* multi-exponentiation P , i.e., we consider a compressed Σ -protocol for the following relation

$$R_{\text{AMORHOM}} = \{ (P, y_1, \dots, y_s; \mathbf{x}) : P = \mathbf{g}^{\mathbf{x}}, y_1 = f_1(\mathbf{x}), \dots, y_s = f_s(\mathbf{x}) \}. \quad (6)$$

This scenario is reduced to the original scenario of opening one homomorphism on one commitment by means of a standard polynomial amortization trick. In the first move of the protocol, the verifier sends a random challenge $\rho \in \mathbb{Z}_q$ to the prover, and then Π_c is executed on the instance given by $P = \mathbf{g}^{\mathbf{x}}$, $f_\rho = f_1 + \rho f_2 + \dots + \rho^{s-1} f_s$ and $y_\rho = y_1 + \rho y_2 + \dots + \rho^{s-1} y_s$.

The core idea behind this construction is the observation that if \mathbf{x} satisfies $f_\rho(\mathbf{x}) = y_\rho$ for s distinct choices of ρ then $f_i(\mathbf{x}) = y_i$ for all $i \in \{1, \dots, s\}$. A caveat is that when trying to extract such an \mathbf{x} by rewinding $s - 1$ times and choosing different ρ 's, one might potentially extract different choices of \mathbf{x} 's. However, since $\mathbf{g}^{\mathbf{x}} = P$ must still hold, this would lead to a non-trivial DL relation among the g_i 's, and thus cannot happen when the prover is computationally bounded.

The properties of this protocol for relation R_{AMORHOM} , denoted by Π_{AMORHOM} , are summarized in Theorem 5. Note that the communication from prover to verifier is identical to that of protocol Π_c . However, the polynomial amortization trick degrades the soundness from unconditional to computational because of the above reason.

Theorem 5 (Amortization over Many Homomorphisms). *Let $n > 2$, then Π_{AMORHOM} is a $(2\mu + 4)$ -move protocol for relation R_{AMORHOM} , where $\mu = \lceil \log_2(n) \rceil - 2$. It is perfectly complete, special honest-verifier zero-knowledge and computationally $(s, 2, 3, 3, \dots, 3)$ -special sound, under the discrete logarithm assumption in \mathbb{G} . Moreover, the communication costs are:*

- $\mathcal{P} \rightarrow \mathcal{V}$: $2 \lceil \log_2(n) \rceil - 3$ elements of \mathbb{G} , $2 \lceil \log_2(n) \rceil - 3$ elements of \mathbb{G}_T and 4 elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: $\lceil \log_2(n) \rceil$ elements of \mathbb{Z}_q .

In the above claim on the computational special soundness we take it as understood that g_1, \dots, g_n are chosen uniformly at random in \mathbb{G} .

Proof. **Completeness** and **SHVZK** follow directly from the corresponding properties of Protocol Π_c .

Special Soundness: From the proof of Theorem 3 we know that for every ρ there exists an efficient algorithm that, from any $(2, 3, \dots, 3)$ -tree (depth $\mu + 1$) of accepting transcripts, extracts a witness \mathbf{z} such that $\mathbf{g}^{\mathbf{z}} = P$ and $f_\rho(\mathbf{z}) = y_1 + \rho y_2 + \dots + \rho^{s-1} y_s$.

We show that there also exists an efficient algorithm that, from s exponents $\mathbf{z}_1, \dots, \mathbf{z}_s \in \mathbb{Z}_q^n$ such that $\mathbf{g}^{\mathbf{z}_i} = P$ and $f_{\rho_i}(\mathbf{z}_i) = y_1 + \rho_i y_2 + \dots + \rho_i^{s-1} y_s$ for all i and for pairwise distinct challenges $\rho_i \in \mathbb{Z}_q$, extracts either a non-trivial DL-relation for the public parameters \mathbf{g} or a witness for relation R_{AMORHOM} . Combining these two results shows that Protocol Π_{AMORHOM} is $(s, 2, 3, \dots, 3)$ -special sound from which knowledge soundness follows from [2].

First suppose that there exist $1 \leq i, j \leq s$ such that $\mathbf{z}_i \neq \mathbf{z}_j$. Then $\mathbf{g}^{\mathbf{z}_i} = P = \mathbf{g}^{\mathbf{z}_j}$ gives a non-trivial DL-relation, which completes the proof for this case.

Now suppose that $\mathbf{z}_i = \mathbf{z}$ for all i . Let $(a_{i,j})_{1 \leq i, j \leq s}$ be the inverse of the Vandermonde matrix generated by the challenges ρ_1, \dots, ρ_s , i.e.,

$$\begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ \rho_1^s & \dots & \rho_s^s \end{pmatrix} \begin{pmatrix} a_{1,1} & \dots & a_{1,s} \\ \vdots & \ddots & \vdots \\ a_{s,1} & \dots & a_{s,s} \end{pmatrix} = I_s.$$

Note that this Vandermonde matrix is invertible because the challenges are pairwise distinct. Then for all $1 \leq i \leq s$ it holds that

$$f_i(\mathbf{z}) = a_{1,i} f_{\rho_1}(\mathbf{z}) + \dots + a_{s,i} f_{\rho_s}(\mathbf{z}) = y_i.$$

Hence \mathbf{z} is a witness for relation R_{AMORHOM} which completes the proof. \square

4 Proving *Partial Knowledge*

Here, we show our new efficient proofs for partial knowledge, i.e., for proving knowledge of k -out-of- n discrete logarithms (Sect. 4.1), and for proving knowledge of k -out-of- n commitment openings (Sect. 4.2). As we will see, these new proofs of partial knowledge follow quite easily by exploiting the core idea of the general construction in [14] and combining it with the techniques and results from the section above. This further demonstrates the strength of combining the compression technique introduced by [6, 9] with general Σ -protocol theory.

4.1 Partial Knowledge of DL's

In this section we construct a simple SHVZK proof of knowledge for proving knowledge of k -out-of- n discrete logarithms. Our protocol inherits the logarithmic communication from the compressed Σ -protocol(s) from the previous section. More precisely, we give a SHVZK protocol for the following relation

$$R_{\text{PARTIAL}} = \left\{ (g, P_1, \dots, P_n \in \mathbb{G}, k \in \{1, \dots, n\}; S \subset \{1, \dots, n\}, \mathbf{x} \in \mathbb{Z}_q^n) : \right. \\ \left. |S| = k, P_i = g^{x_i} \text{ for all } i \in S \right\}. \quad (7)$$

Note that, for notational convenience, the witness \mathbf{x} is defined as a vector in \mathbb{Z}_q^n while only the k coefficients $(x_i)_{i \in S}$ are relevant in this relation.

The protocol goes as follows. First, the prover computes the unique polynomial

$$p(X) = 1 + \sum_{j=1}^{n-k} a_j X^j \in \mathbb{Z}_q[X]$$

of degree at most $n - k$ such that $p(0) = 1$ and $p(i) = 0$ for all $i \notin S$.

Second, the prover computes

$$t_i := p(i)x_i$$

for $i \in \{1, \dots, n\}$ (recall that $p(i)$ vanishes for those i for which the prover does not know x_i), and sends a Pedersen commitment $P \in \mathbb{G}$ to the vector

$$\mathbf{y} = (a_1, \dots, a_{n-k}, t_1, \dots, t_n) \in \mathbb{Z}_q^{2n-k}$$

to the verifier. Here, the commitment P is computed as $P = \mathbf{g}^{\mathbf{y}} h^\gamma$ with respect to public parameters $\mathbf{g} = (g_1, \dots, g_{2n-k}) \in \mathbb{G}^{2n-k}$ and $h \in \mathbb{G}$ for which no non-trivial DL-relations are known to the prover, i.e., so that the commitment is indeed binding.

Finally, the prover proves to the verifier that the committed vector \mathbf{y} satisfies

$$g^{t_i} = P_i^{p(i)} \tag{8}$$

for all $i \in \{1, \dots, n\}$, where the exponent $p(i)$ on the right-hand-side term is understood as the evaluation of the affine function $(w_1, \dots, w_{n-k}) \mapsto 1 + \sum_{j=1}^{n-k} w_j i^j$ applied to a_1, \dots, a_{n-k} . Thus, rewriting (8) as

$$g^{t_i} P_i^{-\sum_j a_j i^j} = P_i \tag{9}$$

we obtain an expression where the left hand side is a group homomorphism f applied to the committed vector \mathbf{y} , and thus the prover can prove one instance of (8) by means of the compressed protocol from Theorem 3; respectively, for improved efficiency, it can invoke the amortized protocol Π_{AMORHOM} from Theorem 5 for proving that (8) holds for all $i \in \{1, \dots, n\}$.

The resulting protocol, denoted Π_{PARTIAL} , is summarized below in Protocol 3. We note that, in line with the amortized protocol it uses as a subroutine, it is *computationally* special sound, based on the assumption that the prover does not know any non-trivial DL-relations among the public parameters. The security and efficiency properties of Π_{PARTIAL} are formally described in Theorem 6.

Theorem 6 (*k-out-of-n SHVZK Proof of Partial Knowledge*). *Let $n > 1$, then Π_{PARTIAL} is a $(2\mu + 5)$ -move protocol for relation R_{PARTIAL} , where $\mu = \lceil \log_2(2n - k + 1) \rceil - 2$. It is perfectly complete, special honest-verifier zero-knowledge and computationally $(n, 2, 3, 3, \dots, 3)$ -special sound, under the discrete logarithm assumption in \mathbb{G} . Moreover, the communication costs are:*

- $\mathcal{P} \rightarrow \mathcal{V}$: $4 \lceil \log_2(2n - k + 1) \rceil - 5$ elements of \mathbb{G} and 4 elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: $\lceil \log_2(2n - k + 1) \rceil$ elements of \mathbb{Z}_q .

Protocol 3. SHVZK Proof of Partial Knowledge Π_{PARTIAL} for Relation R_{PARTIAL}
 Proving knowledge of k -out-of- n discrete logarithms.

PUBLIC PARAMETERS : $\mathbf{g} \in \mathbb{G}^{2n-k}, h \in \mathbb{G}$

INPUT $(g, P_1, \dots, P_n, k; S, \mathbf{x})$

$$S \subset \{1, \dots, n\}, |S| = k$$

$$g^{x_i} = P_i \text{ for } i \in S$$

Prover

Verifier

$$p(X) = 1 + \sum_{i=1}^{n-k} a_i X^i \text{ s.t.}$$

$$p(i) = 0 \quad \forall i \notin S$$

$$\mathbf{y} = (a_1, \dots, a_{n-k},$$

$$p(1)x_1, \dots, p(n)x_n)$$

$$\gamma \leftarrow_R \mathbb{Z}_q, P = \mathbf{g}^{\mathbf{y}} h^\gamma$$

\xrightarrow{P}

Run Π_{AMORHOM} to prove that \mathbf{y} satisfies

$$g^{y_{i+n-k}} P_i^{-\sum_{j=1}^i y_j i^j} = P_i \quad \forall i \in \{1, \dots, n\}$$

Proof. **Completeness** follows in a straightforward manner.

Special Honest Verifier Zero-Knowledge follows immediately from the fact that P is uniformly random and from the corresponding zero-knowledge property of Π_{AMORHOM} .

Special Soundness: The computational special soundness of Π_{AMORHOM} guarantees existence of an extractor that extracts, from any computationally-bounded successful prover, an opening $\mathbf{y} = (a_1, \dots, a_{n-k}, t_1, \dots, t_n)$ of the commitment P for which (9) holds for all $i \in \{1, \dots, n\}$, and thus, considering the corresponding polynomial $p(X) = 1 + \sum_{j=1}^{n-k} a_j X^j$, for which (8) holds for all $i \in \{1, \dots, n\}$. Given the bounded degree of p and the non-zero constant coefficient, $p(i) = 0$ for at most $n - k$ choices of $i \in \{1, \dots, n\}$. Thus, setting $S := \{i : p(i) \neq 0\}$, we have $|S| \geq k$, and for any $i \in S$ we can set $x_i := t_i/p(i)$ and (8) then implies that $g^{x_i} = P_i$. \square

4.2 Partial Knowledge of Commitment Openings

In the previous section we constructed a protocol for proving knowledge of k -out-of- n discrete logarithms or, equivalently, a protocol for showing that a prover can open k -out-of- n Pedersen commitments to 0. This protocol can easily be adapted to accommodate, for example, the following variation of this zero-knowledge scenario.

In this variation we let P_1, \dots, P_n be Pedersen commitments, for which the prover claims to know k -out-of- n openings, not necessarily to 0. More precisely, the prover claims to know a witness for the following relation:⁵

$$R_{\text{PARTIALCOM}} = \left\{ (g, h, P_1, \dots, P_n \in \mathbb{G}, k \in \{1, \dots, n\}; S \subset \{1, \dots, n\}, \right. \\ \left. x_1, \dots, x_n \in \mathbb{Z}_q, \gamma_1, \dots, \gamma_n \in \mathbb{Z}_q) : \right. \\ \left. |S| = k, P_i = g^{x_i} h^{\gamma_i} \text{ for all } i \in S \right\}. \quad (10)$$

A proof of knowledge for relation $R_{\text{PARTIALCOM}}$ is obtained by applying the following adaptations. After defining the polynomial $p(X)$ as before, the prover computes

$$t_i := p(i)x_i \in \mathbb{Z}_q \quad \text{and} \quad s_i := p(i)\gamma_i \in \mathbb{Z}_q,$$

for $i \in \{1, \dots, n\}$ and sends a Pedersen commitment $P \in \mathbb{G}$ to the vector

$$\mathbf{y} = (a_1, \dots, a_{n-k}, t_1, \dots, t_n, s_1, \dots, s_n) \in \mathbb{Z}_q^{3n-k},$$

to the verifier. Finally, by invoking Protocol Π_{AMORHOM} , the prover shows that

$$g^{t_i} h^{s_i} P_i^{-\sum_j a_j i^j} = P_i$$

for all $i \in \{1, \dots, n\}$. Formally, we have the following security and efficiency properties.

Theorem 7 (k -out-of- n SHVZK Proof of Partial Knowledge for Commitment Openings). *$\Pi_{\text{PARTIALCOM}}$ is a $(2\mu + 5)$ -move protocol for relation $R_{\text{PARTIALCOM}}$, where $\mu = \lceil \log_2(3n - k + 1) \rceil - 2$. It is perfectly complete, special honest-verifier zero-knowledge and computationally $(n, 2, 3, 3, \dots, 3)$ -special sound, under the discrete logarithm assumption in \mathbb{G} . Moreover, the communication costs are:*

- $\mathcal{P} \rightarrow \mathcal{V}$: $4 \lceil \log_2(3n - k + 1) \rceil - 5$ elements of \mathbb{G} and 4 elements of \mathbb{Z}_q .
- $\mathcal{V} \rightarrow \mathcal{P}$: $\lceil \log_2(3n - k + 1) \rceil$ elements of \mathbb{Z}_q .

Remark 2. We emphasize that $\Pi_{\text{PARTIALCOM}}$ is only special sound under the assumption that the prover does not know a non-trivial DL relation between the public parameters $\mathbf{g} \in \mathbb{G}^{3n-k}$ and $h \in G$ for the Pedersen commitment P to the vector \mathbf{y} , i.e., it is crucial that the commitment P is binding. By contrast, the special soundness of $\Pi_{\text{PARTIALCOM}}$ does not depend on a computational assumption regarding the public parameters $g, h \in \mathbb{G}$ for the Pedersen commitments P_i , i.e., the commitments P_i are not required to be binding for Protocol $\Pi_{\text{PARTIALCOM}}$ to be special sound.

5 Extensions and Generalizations

Our techniques from Sect. 4 for proofs of partial knowledge can be extended and generalized in various directions. We discuss some examples here.

⁵ The element $h \in \mathbb{G}$, used in the commitments P_i , is not necessarily the same element as the element $h \in \mathbb{G}$ used in the Pedersen vector commitment P of Protocol Π_{PARTIAL} .

5.1 Pairing Based Commitments to Reduce the Communication Complexity

We show here that by introducing a pairing and considering a pairing based extension of Pedersen’s vector commitment scheme (see below), we can incorporate a trick from [9] to reduce the relevant constant by another factor up to 2.

Recall that, rather than a general homomorphism $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T$, [2] considers the special case of a *linear form* $L : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$, with the goal to prove that a secret vector $\mathbf{x} \in \mathbb{Z}_q^n$, committed to as $P = \mathbf{g}^{\mathbf{x}}h^\gamma$, satisfies $L(\mathbf{x}) = y$ for a publicly known P and y . The trick then is to include y into the commitment by considering $P' = \mathbf{g}^{\mathbf{x}}h^\gamma k^y$ instead, and proving it to be of the claimed form using a Σ -protocol and then compressing it. The gained advantage is not that y becomes hidden in the commitment— y is still known, and P' would actually be computed by the verifier from P and y —but that the public information is reduced to a single group element. In the language of our general view (Appendix A), Protocol 4 is applied to the homomorphism $\mathbb{Z}_q^n \rightarrow \mathbb{G}_T$, $\mathbf{x} \mapsto \mathbf{g}^{\mathbf{x}}h^\gamma k^{L(\mathbf{x})}$, rather than to $\mathbb{Z}_q^n \rightarrow \mathbb{G}_T \times \mathbb{Z}_q$, $\mathbf{x} \mapsto (\mathbf{g}^{\mathbf{x}}h^\gamma, L(\mathbf{x}))$. Thereby, in every recursion of the compression mechanism, each “cross term” consists of just one element in \mathbb{G}_T now, rather than a pair in $\mathbb{G}_T \times \mathbb{Z}_q$. Overall this reduces the communication costs by roughly a factor up to 2, depending on the choice of the group \mathbb{G}_T and the representation of its elements.

To apply this approach to our scenario, and incorporate $f(\mathbf{x}) \in \mathbb{G}_T$ into the commitment, we require a compact vector commitment scheme for vectors $(\mathbf{x}, y) \in \mathbb{Z}_q^n \times \mathbb{G}_T$, which have coefficients in both \mathbb{Z}_q and \mathbb{G}_T . Under bilinear pairing assumptions these commitment schemes exist [1, 25]. Namely, let us assume that there exists a group \mathbb{G}_2 of prime order q , and a bilinear pairing $e : \mathbb{G}_T \times \mathbb{G}_2 \rightarrow \mathbb{G}$. For public parameters $\mathbf{g} \in \mathbb{G}^n$, $h \in \mathbb{G}$ and $R \in \mathbb{G}_2$ sampled uniformly at random, we can define the following commitment scheme:

$$\text{COM}' : \mathbb{Z}_q^n \times \mathbb{G}_T \times \mathbb{Z}_q \rightarrow \mathbb{G}, \quad (\mathbf{x}, y, \gamma) \mapsto \mathbf{g}^{\mathbf{x}}h^\gamma e(y, R), \quad (11)$$

where $\gamma \in \mathbb{Z}_q$ is chosen uniformly at random to commit to an element $(\mathbf{x}, y) \in \mathbb{Z}_q^n \times \mathbb{G}_T$. This commitment scheme is unconditionally hiding and binding under the assumption that the prover does not know a non-zero vector $(\mathbf{x}, y, \gamma) \in \mathbb{Z}_q^n \times \mathbb{G}_T \times \mathbb{Z}_q$ such that $\mathbf{g}^{\mathbf{x}}h^\gamma e(y, R) = 1 \in \mathbb{G}$. This assumption is implied by the double pairing (DBP) assumption, which is in turn implied by the decisional Diffie-Hellman assumption over \mathbb{G}_T [1, 25].

A more efficient protocol for opening arbitrary homomorphisms $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T$ is now obtained by replacing the Pedersen vector commitment scheme by this pairing based commitment scheme that allows the group element $f(\mathbf{x})$ to be incorporated into the commitment. The resulting compressed Σ -protocol for opening homomorphisms is derived as in Sect. 3, but with the generic compression Protocol 4 now instantiated with the group homomorphism $\mathbb{Z}_q^{n+1} \rightarrow \mathbb{G}$, $(\mathbf{x}, \gamma) \mapsto \mathbf{g}^{\mathbf{x}}h^\gamma e(cf(\mathbf{x}), R)$, for a random challenge $c \in \mathbb{Z}_q$, rather than $\mathbb{Z}_q^{n+1} \rightarrow \mathbb{G} \times \mathbb{G}_T$, $(\mathbf{x}, \gamma) \mapsto (\mathbf{g}^{\mathbf{x}}h^\gamma, f(\mathbf{x}))$. Applying this modification to the k -out-of- n proof

of partial knowledge (Protocol 3) results in communication costs, from prover to verifier, of exactly $2 \lceil \log_2(2n - k + 1) \rceil - 1$ elements of \mathbb{G} and 4 elements of \mathbb{Z}_q .

5.2 Multi-exponentiations and Vector Commitments

A straightforward generalization of Protocol Π_{PARTIAL} shows that, instead of the DL problem for standard exponentiations, we can also consider multi-exponentiations. More concretely, this generalization gives a protocol for the following relation

$$R' = \left\{ (\mathbf{h} \in \mathbb{G}^m, P_1, \dots, P_n \in \mathbb{G}, k \in \{1, \dots, n\}; S \subset \{1, \dots, n\}, \mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{Z}_q^m) : |S| = k, P_i = \mathbf{h}^{\mathbf{x}_i} \text{ for all } i \in S \right\}. \quad (12)$$

The only adaptation of protocol Π_{PARTIAL} that is required is the replacement of the scalars $x_i \in \mathbb{Z}_q$ by vectors $\mathbf{x}_i \in \mathbb{Z}_q^m$. The communication complexity of the resulting protocol grows logarithmically in the dimension m of the multi-exponentiations. In a completely analogous manner, protocol $\Pi_{\text{PARTIALCOM}}$ from Sect. 4.2 can be generalized to proving partial knowledge of Pedersen vector commitment openings.

5.3 Plug and Play with Circuit Zero-Knowledge

In many practical scenarios, one wishes to prove not only partial knowledge of commitment openings, but also that the committed values satisfy some additional constraints. Typically these constraints are defined by an arithmetic circuit $C : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ and the committed values $x_1, \dots, x_n \in \mathbb{Z}_q$ are claimed to satisfy $C(x_1, \dots, x_n) = 0$. More concretely, we consider a prover that claims to know a witness for the following relation

$$R_{\text{PARTIALCIRC}} = \left\{ (g, h, P_1, \dots, P_n \in \mathbb{G}, k \in \{1, \dots, n\}; S \subset \{1, \dots, n\}, x_1, \dots, x_n \in \mathbb{Z}_q, \gamma_1, \dots, \gamma_n \in \mathbb{Z}_q) : |S| = k, C(x_1, \dots, x_n) = 0, P_i = g^{x_i} h^{\gamma_i} \text{ for all } i \in S \right\}. \quad (13)$$

Note that in this relation the prover is only committed to k -out-of- n scalars x_i , i.e., it can choose $n - k$ scalars freely.

To handle this extension of the partial knowledge scenario we deploy the circuit ZK techniques from [2]. For these techniques to be applicable all we need to show is that we can open homomorphisms and linear forms on the same Pedersen vector commitment. In [2] it is namely shown how circuit ZK protocols, for arbitrary arithmetic circuits, are derived from the functionality of opening linear forms on Pedersen vector commitments.

However, for any homomorphism $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T$ and any linear form $L : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ it is easily seen that the following map is again a homomorphism

$$(f, L) : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T \times \mathbb{Z}_q \quad \mathbf{x} \mapsto (f(\mathbf{x}), L(\mathbf{x})).$$

So the functionality of Protocol Π_c , opening homomorphisms, trivially extends to the functionality of opening homomorphisms *and* linear forms on the same vector commitment.

Applying this approach directly results in a protocol for relation $R_{\text{PARTIALCIRC}}$ where the communication costs, from prover to verifier, are roughly $6 \log_2(n)$ elements. These communication costs can be reduced to roughly $4 \log_2(n)$ elements, or $2 \log_2(n)$ on a pairing based platform, by applying the techniques from Sect. 5.1 and [2].

Remark 3. Various other (natural) circuit ZK scenarios exist. For example, when the circuit $C : \mathbb{Z}_q^k \rightarrow \mathbb{Z}_q$ only takes the scalars x_i for $i \in S$ as input. Many of these scenarios are easily dealt with by plug and play (modular design) with the techniques from [2].

5.4 General Access Structures

Thus far, we have restricted ourselves to provers that claim to know the solutions of some (secret) subset S , of cardinality at least k , of n (public) DL problems $P_i = g^{x_i}$, i.e., the secret subset S is an element of a *threshold* access structure

$$\Gamma_{k,n} = \{A \subset \{1, \dots, n\} : |A| \geq k\} \subset 2^{\{1, \dots, n\}}.$$

Here, we describe how the protocols from Sect. 4 can easily be generalized to arbitrary monotone access structures $\Gamma \subset 2^{\{1, \dots, n\}}$, i.e., to provers that claim to know the solutions of some subset of $S \in \Gamma$ of n DL problems. Recall that Γ is called a monotone access structure if for all $A \in \Gamma$ and for all $B \subset 2^{\{1, \dots, n\}}$ with $A \subset B$ it holds that $B \in \Gamma$. The proofs of partial knowledge of [14] already considered arbitrary access structures and we adapt their techniques by combining them with our compression framework.

Our proofs of k -out-of- n partial knowledge implicitly deploy a linear secret sharing scheme (LSSS) for access structure $\Gamma_{k,n}^* = \Gamma_{n-k,n}$. Here, Γ^* denotes the *dual* of access structure Γ , generally given by

$$\Gamma^* = \{A \subset \{1, \dots, n\} : A^c \notin \Gamma\}.$$

More concretely the protocols of Sect. 4 use Shamir's secret sharing scheme and the polynomial $p(X) = 1 + \sum_{j=1}^{n-k} a_j X^j$ defines a secret sharing of the field element 1.

To construct a proof of partial knowledge for monotone access structure Γ we simply replace $p(i)$ by the i -th share (which may consist of several field elements, depending on the expansion factor) of a linear secret sharing of 1, with the randomness chosen so that the “right” shares (i.e., those corresponding to the x_i 's that the prover does not know) vanish.

Note that an honest prover knows $(x_i)_{i \in S}$ for some $S \in \Gamma$. Hence, $S^c \notin \Gamma^*$ and for this reason the appropriate secret sharing of 1 exists, showing completeness of the generalized proof of partial knowledge.

Special soundness follows from the following observation. Let $A \subset \{1, \dots, n\}$ be the subset for which all the corresponding shares vanish. Then, by linearity of the secret sharing scheme and since the secret sharing reconstructs to 1, it follows that $A \notin \Gamma^*$. Hence, $A^c \in \Gamma$ and special soundness follows as before.

The communication complexity of the resulting protocol depends logarithmically on the size of the LSSS for Γ^* , which is given by the monotone-span-program complexity of Γ^* [32] and which coincides with the monotone-span-program complexity of Γ [17].

Acknowledgements. Thomas Attema has been supported by EU H2020 project No 780701 (PROMETHEUS) and by the Vraaggestuurd Programma Veilige Maatschappij, supervised by the Innovation Team of the Dutch Ministry of Justice and Security. Ronald Cramer has been supported by ERC ADG project No 74079 (ALGSTRONGCRYPTO) and by the NWO Gravitation Programme (QSC).

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. *J. Cryptol.* **29**(2), 363–421 (2016)
2. Attema, T., Cramer, R.: Compressed Σ -protocol theory and practical application to plug & play secure algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) *CRYPTO 2020*. LNCS, vol. 12172, pp. 513–543. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_18
3. Attema, T., Cramer, R., Kohl, L.: A compressed Σ -protocol theory for lattices. In: *CRYPTO (2021)*, to appear
4. Benaloh, J.C., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Hellese, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_24
5. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) *ESORICS 2015*. LNCS, vol. 9326, pp. 243–265. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_13
6. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) *EUROCRYPT 2016*. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_12
7. Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) *CRYPTO 2002*. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_30
8. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: towards privacy in a smart contract world. In: Bonneau, J., Heninger, N. (eds.) *FC 2020*. LNCS, vol. 12059, pp. 423–443. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_23
9. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: *IEEE Symposium on Security and Privacy*, pp. 315–334. IEEE Computer Society (2018)
10. Camenisch, J.: Efficient and generalized group signatures. In: Fumy, W. (ed.) *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 465–479. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_32

11. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
12. Cramer, R.: Modular design of secure yet practical cryptographic protocols. Ph.D. thesis, CWI and University of Amsterdam (1996)
13. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055745>
14. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
15. Diamond, B.E.: “Many-out-of-many” proofs with applications to anonymous Zether. In: IEEE Symposium on Security and Privacy. IEEE (2021)
16. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
17. Gál, A.: Combinatorial methods in Boolean function complexity. Ph.D. thesis, University of Chicago (1995)
18. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: STOC, pp. 291–304. ACM (1985)
19. Goodell, B., Noether, S.: Thring signatures and their applications to spender-ambiguous digital currencies. IACR Cryptol. ePrint Arch. **2018**, 774 (2018)
20. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_19
21. 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. LNCS, vol. 9057, pp. 253–280. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_9
22. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash Protocol Specification - Version 2020.1.7 (2020). <https://github.com/zcash/zips/blob/master/protocol/protocol.pdf>
23. Jivanyan, A.: Lelantus: towards confidentiality and anonymity of blockchain transactions from standard assumptions. IACR Cryptol. ePrint Arch. **2019**, 373 (2019)
24. Jivanyan, A., Mamikonyan, T.: Hierarchical one-out-of-many proofs with applications to blockchain privacy and ring signatures. In: AsiaJCIS, pp. 74–81. IEEE (2020)
25. Lai, R.W.F., Malavolta, G., Ronge, V.: Succinct arguments for bilinear group arithmetic: practical structure-preserving cryptography. In: ACM Conference on Computer and Communications Security, pp. 2057–2074. ACM (2019)
26. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptol. **16**(3), 143–184 (2003)
27. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy, pp. 122–134. IEEE Computer Society (1980)
28. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: IEEE Symposium on Security and Privacy, pp. 397–411. IEEE Computer Society (2013)

29. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
30. Sander, T., Ta-Shma, A., Yung, M.: Blind, auditable membership proofs. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 53–71. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45472-1_5
31. Serhack: Mastering Monero: The Future of Private Transactions. Independently Published (2020)
32. Simmons, G.J., Jackson, W.A., Martin, K.M.: The geometry of shared secret schemes. Bull. Inst. Combin. Appl. **1**, 71–88 (1991)

A General View on the Compression

We consider here the natural generalization of the compression Protocol 2 to an arbitrary group homomorphism $\Psi : \mathbb{H} \rightarrow \mathbb{G}$ for groups \mathbb{H} and \mathbb{G} of prime exponent⁶ q and for which \mathbb{H} is a direct sum $\mathbb{H} = \mathbb{H}' \oplus \mathbb{H}'$ of a group \mathbb{H}' with itself. Thus, any $x \in \mathbb{H}$ can be written as a tuple $x = (x_L, x_R)$ of group elements $x_L, x_R \in \mathbb{H}'$. By convention, we write \mathbb{H}' , and thus \mathbb{H} , as an additive group and \mathbb{G} as a multiplicative group. Protocol 4, denoted by Π_Ψ , below is a proof of knowledge for the relation

$$R_\Psi = \{(P; x) \in \mathbb{G} \times \mathbb{H} : \Psi(x) = P\}.$$

Its properties are summarized in the following theorem. The proof is along the very same lines as the proof of Theorem 2, with obvious adjustments. We provide it here for completeness.

Theorem 8 (General Compression Mechanism). *Let $\mathbb{H} = \mathbb{H}' \oplus \mathbb{H}'$ for some group \mathbb{H}' . Then Π_Ψ is a 3-move protocol for relation R_Ψ . It is perfectly complete and unconditionally 3-special sound. Moreover, the communication costs are:*

- $\mathcal{P} \rightarrow \mathcal{V}$: 2 elements of \mathbb{G} and 1 element of \mathbb{H}' .
- $\mathcal{V} \rightarrow \mathcal{P}$: 1 element of \mathbb{Z}_q .

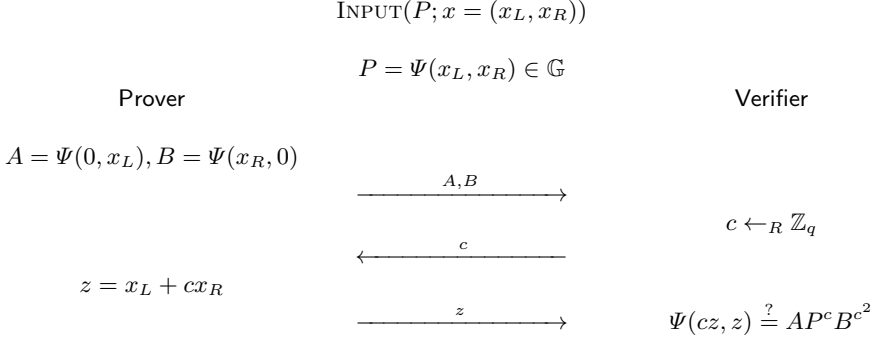
Proof. **Completeness** follows directly.

Special Soundness: We show that the protocol is 3-special sound. Let (A, B, c_1, z_1) , (A, B, c_2, z_2) and (A, B, c_3, z_3) be three accepting transcripts for distinct challenges $c_1, c_2, c_3 \in \mathbb{Z}_q$. Let $a_1, a_2, a_3 \in \mathbb{Z}_q$ be such that

$$\begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^2 & c_2^2 & c_3^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Note that, since the challenges are distinct, this Vandermonde matrix is invertible and a solution to this equation exists. We define $\bar{z} = \sum_{i=1}^3 a_i (c_i z_i, z_i)$ for which it is easily verified that $\Psi(\bar{z}) = P$. Hence, \bar{z} is a witness for relation R_Ψ , which completes the proof. □

Protocol 4. Generic Compression Mechanism II_Ψ for relation R_Ψ .



Considering the setting of Sect. 3 and instantiating Ψ with $\Psi : \mathbb{Z}_q^n \rightarrow \mathbb{G} \times \mathbb{G}_T$, $\mathbf{x} \mapsto (\mathbf{g}^{\mathbf{x}}, f(\mathbf{x}))$ for the considered group homomorphism $f : \mathbb{Z}_q^n \rightarrow \mathbb{G}_T$, with n assumed to be even so that $\mathbb{Z}_q^n = \mathbb{Z}_q^{n/2} \oplus \mathbb{Z}_q^{n/2}$, we recover the relation R_f and Protocol 2 from Sect. 3. Similarly, we recover the pairing-based compression protocol of Sect. 5.1 by instantiating Ψ with $\Psi : \mathbb{Z}_q^n \rightarrow \mathbb{G}$, $\mathbf{x} \mapsto \mathbf{g}^{\mathbf{x}} e(f(\mathbf{x}), R)$.

Consider the final verification $\Psi(cz, z) \stackrel{?}{=} AP^c B^{c^2}$ in Protocol 4. In line with Protocol 2 in Sect. 3, when we define, for an arbitrary given $c \in \mathbb{Z}_q$, the group homomorphism $\Psi' : \mathbb{H}' \rightarrow \mathbb{G}$, $z \mapsto \Psi(cz, z)$ and the group element $P' = AP^c B^{c^2}$, we observe that the final verification step in Protocol 4 is to check if (P, z) satisfies the relation $R_{\Psi'}$. Therefore, if \mathbb{H}' happens to again be a direct sum $\mathbb{H}' = \mathbb{H}'' \oplus \mathbb{H}''$ of a group \mathbb{H}'' with itself, we can replace the last communication and verification step in Protocol 2 by an execution of Protocol 2 for the relation $R_{\Psi'}$. Thus, if \mathbb{H} is actually the n -fold direct sum of a group \mathbb{H}_\circ with itself for n a power of 2 (which we may assume without loss of generality), we obtain a proof of knowledge for relation R_Ψ , where the communication costs, from prover to verifier, are $2 \log(n)$ elements of \mathbb{G} and 1 element of \mathbb{H}_\circ .

⁶ Recall that the exponent of group is the least common multiple of the orders of all group elements, i.e., it is the smallest e such that $g^e = 1$ for all group elements g .



Mac'n'Cheese: Zero-Knowledge Proofs for Boolean and Arithmetic Circuits with Nested Disjunctions

Carsten Baum^{1(✉)}, Alex J. Malozemoff², Marc B. Rosen², and Peter Scholl¹

¹ Aarhus University, Aarhus, Denmark
cbaum@cs.au.dk

² Galois, Inc., Portland, USA

Abstract. Zero knowledge proofs are an important building block in many cryptographic applications. Unfortunately, when the proof statements become very large, existing zero-knowledge proof systems easily reach their limits: either the computational overhead, the memory footprint, or the required bandwidth exceed levels that would be tolerable in practice.

We present an interactive zero-knowledge proof system for boolean and arithmetic circuits, called *Mac'n'Cheese*, with a focus on supporting large circuits. Our work follows the commit-and-prove paradigm instantiated using information-theoretic MACs based on vector oblivious linear evaluation to achieve high efficiency. We additionally show how to optimize disjunctions, with a general OR transformation for proving the disjunction of m statements that has communication complexity proportional to the longest statement (plus an additive term logarithmic in m). These disjunctions can further be *nested*, allowing efficient proofs about complex statements with many levels of disjunctions. We also show how to make *Mac'n'Cheese* non-interactive (after a preprocessing phase) using the Fiat-Shamir transform, and with only a small degradation in soundness.

We have implemented the online phase of *Mac'n'Cheese* and achieve a runtime of 144 ns per AND gate and 1.5 μ s per multiplication gate in $\mathbb{F}_{2^{61}-1}$ when run over a network with a 95 ms latency and a bandwidth of 31.5 Mbps. In addition, we show that the disjunction optimization improves communication as expected: when proving a boolean circuit with eight branches and each branch containing roughly 1 billion multiplications, *Mac'n'Cheese* requires only 75 more bytes to communicate than in the single branch case.

1 Introduction

Zero knowledge (ZK) proofs are interactive protocols which allow a prover P to convince a verifier V that a certain statement x is true in such a way that V learns nothing beyond the validity of the statement. ZK proofs have a wide range of applications in cryptography, from signatures [BG90] to compiling other protocols from passive to active security [GMW87]. More recently, ZK proofs have

seen widespread applications outside of classical cryptography, for example in the cryptocurrency space [BCG+14]. These constructions mostly focus on *succinctness* and *non-interactivity*; namely, the construction of “succinct” proofs that have a small verification runtime and that do not require interaction between P and V for validation.

However, for sufficiently large statements—on the order of billions of instructions—most existing proof systems fail due to either memory constraints or high prover running times. Systems such as SNARKs [BCG+13] or recent IOP-based constructions such as Ligerio [AHIV17] or STARKs [BBHR19] suffer from exactly this drawback: they have an inherent asymptotic prover overhead, paying at least a multiplicative factor $\log(|x|)$ in computation when the statement has length $|x|$, and they need to keep the entire statement x in memory.

1.1 Our Approach: Mac’n’Cheese

In this work we introduce a family of novel ZK proof protocols called Mac’n’Cheese, which are optimized for statements at scale. We use the *commit-and-prove* paradigm [CD97], where we “commit to” values using an information-theoretic message authentication code (MAC). For each committed value, P holds the MAC’ed value and the tag, and V holds the MAC key. Such commitments can be generated very efficiently using vector oblivious linear evaluation (VOLE) [BCGI18] in a preprocessing phase, which can generate many such random commitments with only a small amount of interaction and computation [WYKW20].

Naively, this commit-and-prove approach leads to a proof with bandwidth costs that scale linearly with the circuit size. To decrease this, in Mac’n’Cheese we support efficiently evaluating *disjunctive statements*, namely, to prove that one out of m statements is true, the prover only needs to communicate the information needed to evaluate the true branch among all m disjunctions. Both parties still perform the computations necessary to evaluate each branch, but the verifier uses the messages for the correct branch for all m instances simultaneously. The idea of optimizing disjunctive statements in this way was first considered in recent work on stacked garbling [HK20b], with proofs of disjunctions based on garbled circuits. They observed that disjunctive statements can arise in many natural applications, such as when proving in zero-knowledge the existence of a bug in a program, so optimizing these is well-motivated.

At a high level, our technique can be seen as a generalized OR composition for m protocols, where the resulting OR proof has communication complexity proportional to $\max\{C_i\}$, where C_i is the complexity of the i -th protocol. Contrasted with the classic OR proof approach [CDS94], which requires $\sum C_i$ communication, our techniques for stacking save a multiplicative factor of m . On the other hand, compared with stacked garbling [HK20b], our underlying protocols have around $20\times$ less communication, and are also more flexible, since we can support both boolean and arithmetic circuits.

Efficiency comparison and related work. Table 1 shows the efficiency of our protocols alongside other VOLE- or garbled-circuit-based protocols,

Table 1. Comparison of different GC and VOLE-based ZK protocols (costs exclude OT/VOLE setup). “Comm.” denotes the number of field elements communicated per multiplication. “Rounds” denotes the total number of rounds required, where we count rounds as the number of message flows, so one round is a single message from the prover to the verifier. “Mmps” denotes the number of multiplication gates per second in millions. *We caution against reading too much into these numbers due to differing experimental environments, and provide them mostly as a rough comparison guide.* “Disjunctions” denotes those protocols that support communication-optimized disjunctions. The variable b denotes the batch-size of multiplications, and ϵ denotes a value close to zero that depends on b . Concretely, for a batch size of $b = 1\,000\,000$ we require 17 rounds with $\epsilon = .008$ for the boolean case (using $\mathbb{F}_{2^{40}}$) and $\epsilon = .257$ for the arithmetic case (using \mathbb{F}_p for $p = 2^{61} - 1$).

Protocol	Boolean			Arithmetic			Disjunctions
	Comm.	Rounds	Mmps	Comm.	Rounds	Mmps	
Stacked garbling [HK20b]	128	3	0.3 ^a	—	—	—	✓
Wolverine [WYKW20]	7*	3	2.0 ^b	4	3	0.2 ^b	✗ [†]
Line-Point ZK [DIO21]	—	—	—	1	3	—	✗
QuickSilver [YSWW21]	1	3	12.2 ^c	1	3	1.4 ^c	✗
Mac’n’Cheese (simple)	9	3	—	3	3	—	✓
Mac’n’Cheese (batched)	$1 + \epsilon^*$	$O(\log b)$	6.9 ^d	$1 + \epsilon^*$	$O(\log b)$	0.6 ^d	✓

* For large batches (e.g., $b \geq 1$ million).

† While we believe *Wolverine* can be combined with our approach described in Sect. 3.1, the performance implications of this combination are unclear.

^a With a 100 ms latency and a 100 Mbps bandwidth.

^b With a 0.1 ms latency and a 50 Mbps bandwidth.

^c With a 0.1 ms latency and a 30 Mbps bandwidth for boolean and a 100 Mbps bandwidth for arithmetic.

^d With a 93 ms latency and a 31.5 Mbps bandwidth.

where we focus on communication cost per multiplication gate measured in field elements. As far as we are aware, there are only two ZK approaches that can successfully scale to large statements: the garbled circuit ZK approach [JKO13,FNO15,ZRE15,HK20b], and the more recent approach based on VOLE, namely the concurrent works *Wolverine* [WYKW20] and *Line-Point ZK* [DIO21], plus *QuickSilver* [YSWW21] (which builds on *Line-Point ZK*). All of these VOLE-based approaches have provers that run linear in the proof statement alongside the ability to “stream”—namely, the prover and verifier are not required to store the entire proof statement in memory.

For *Mac’n’Cheese*, our first class of “simple” protocols reduces the communication complexity of *Wolverine* from 4 to 3 field elements for arithmetic circuits, and achieves a slightly higher cost (9 bits) for boolean circuits, while avoiding the need to amortize over many gates. Our second set of protocols has essentially the same practical cost as *Line-Point ZK* and *QuickSilver*, but is best run in large batches, so suited for bigger circuits. Importantly, all of our protocols

are compatible with our technique for efficient disjunctions—we currently do not know how to efficiently adapt this technique for QuickSilver or Line-Point ZK¹.

Finally, note that for zero knowledge from garbled circuits, the best approach currently has a communication cost of 128 bits per AND gate, which is around $18\times$ higher than our approach that also supports disjunctive statements. We note that Heath and Kolesnikov [HK20b] were the first to consider “stacked” disjunctive proofs. Our approach was inspired by stacked garbling, but the technique is very different, and closer in spirit to the earlier ‘free if’ for private function evaluation [Kol18] (although neither technique follows from the other).

Implementation. We implemented the online phase of Mac’n’Cheese in the Rust programming language. Currently, we do not have an implementation of VOLE, which should add a small amount of communication—0.42 bits per VOLE—and slight increase in runtime—at most 85 ns per VOLE [WYKW20, Table 4].

When run on a real-world network (95 ms latency and a bandwidth of 31.5 Mbps), Mac’n’Cheese requires approximately 1.5 μs per multiplication gate (for $\mathbb{F}_{2^{61}-1}$), and 144 ns per AND gate (using $\mathbb{F}_{2^{40}}$). Run locally, Mac’n’Cheese requires approximately 276 ns per multiplication gate and 141 ns per AND gate. We also show that disjunctions have large communication savings: when run on a circuit containing eight branches each of which contains 1 billion multiplication gates, we see a communication increase of only 75 bytes versus running a single 1-billion-gate branch.

1.2 Our Techniques

We present the Mac’n’Cheese approach in four steps: first, we describe the zero-knowledge protocol in a setting with idealized homomorphic commitments to single field elements. Next, we present an abstraction for such protocols which we call *Interactive Protocols with Linear Oracle Verification—IPs with LOVE* for short—and explain how IPs with LOVE naturally support nested disjunctions and can be compiled to ZK protocols using VOLE. We then provide efficient IPs with LOVE for general circuit satisfiability, which intuitively follow from such protocols for homomorphic commitments. Finally, we show that our protocols are compatible with streaming and that we can apply the Fiat-Shamir transform to reduce the round complexity with only a small loss in soundness.

Circuit satisfiability via idealized homomorphic commitments. Assume that the statement x , together with a witness \mathbf{w} , is provided to P while V only obtains x . We consider x as a circuit C over a finite field \mathbb{F} , such that $C(\mathbf{w}) = 0$ iff $(x, \mathbf{w}) \in \mathcal{R}$ and assume that \mathbf{w} is a vector over \mathbb{F} .

¹ The challenge in applying our disjunction optimization to these protocols is that the verification check requires input from V , which allows a malicious V to try to guess the evaluated branch by supplying an invalid value for all other branches.

Implementing the test that $C(\mathbf{w}) = 0$ can be done using standard techniques with idealized homomorphic commitments [CD98], but we nevertheless sketch these now. First, P commits to (1) \mathbf{w} , (2) triples of the form a, b, c such that $c = a \cdot b$, and (3) the outputs of all the gates of $C(\mathbf{w})$. P and V then engage in an interactive protocol to test that:

1. The commitments to gate outputs are consistent with C and \mathbf{w} ; and
2. The output of the output gate of C is zero.

Note that these checks reduce to testing that certain committed values are zero:

- This is clear for testing the output of the output gate.
- For each addition gate (or multiplications with public constants from \mathbb{F}) one can simply apply the respective linear operation to the commitments to the inputs of the gate, subtract the commitment of the output and test if the result is a commitment to zero.
- For each multiplication gate, we use Beaver’s circuit randomization approach [Bea92, CD98, KOS16] to reduce multiplication to zero-testing a commitment to a linear combination of commitments to the gate inputs, outputs, and the random triples (a, b, c) , alongside an additional random element sent by V . (In fact, this random element can be generated by the output of a random oracle on the protocol transcript using the Fiat-Shamir transform. We provide more details on this in Sect. 4.1.)

When instantiating homomorphic commitments with VOLE (as we describe later), this basic protocol has an amortized communication complexity of 3 field elements per multiplication gate. This improves upon the arithmetic protocol of Weng et al. [WYKW20], which uses 4 field elements, although they also present a variant with 2 field elements per multiplication which has a higher computational cost due to polynomial operations.

Formalizing security using IPs with LOVE (Sect. 2). Proofs based on ideal homomorphic commitments can be modeled as a functionality where the prover initially commits to some secret values, and the verifier is then allowed to perform linear queries to the commitments, to check that certain relations hold. For instance, linear interactive oracle proofs (IOPs) [BBC+19] model exactly this. In Sect. 2, we extend this paradigm with a new abstraction called *interactive proofs with linear oracle verification (IPs with LOVE)*. In this abstraction, P begins by committing some proof string $\boldsymbol{\pi}$ to an oracle \mathcal{O} . The parties then exchange messages for a fixed number of rounds, after which V sends multiple queries of the form (\mathbf{z}_i, y_i) to \mathcal{O} . These queries are determined by V based on the messages that it received in the previous rounds. \mathcal{O} truthfully tells V if $\langle \boldsymbol{\pi}, \mathbf{z}_i \rangle = y_i$ or not for each of these queries. Eventually, V outputs a bit to represent whether it accepts or not.

The key difference between IPs with LOVE and linear IOPs [BBC+19], is that on top of oracle queries, we allow the prover and verifier to exchange a number of messages. Therefore, IPs with LOVE naturally model homomorphic

commitments in the same way as linear IOPs, while also giving extra power from the exchange of messages, which is what we exploit in our protocols for efficient disjunctions.

From IPs with LOVE to IPs with VOLE. We show that any Public Coin IP with LOVE can be combined with a VOLE protocol to obtain a ZK proof. This is described in Sect. 2.2. We instantiate the oracle \mathcal{O} that contains the string π using information-theoretic commitments (or MACs) of the form

$$\text{MAC}_{(\alpha,\beta)}(x) := x\alpha + \beta,$$

where x comes from a field \mathbb{F}_p and all remaining value from an extension field \mathbb{F}_{p^k} for $k \geq 1$. We call α the “MAC key” and β the “MAC offset”, and sometimes use the notation K to denote the tuple (α, β) , held by the verifier, and τ to denote the MAC tag, held by the prover. These commitments are linearly homomorphic for keys that share the α component, so we can realize each oracle query as a zero-test on such commitments. Their binding guarantee follows from the size of \mathbb{F}_{p^k} .

A batch of n MACs on random values is exactly equivalent to a VOLE of length n , since the MAC relation can be viewed as evaluating a linear function on the input x . This can be generated with high efficiency using recent (random) VOLE protocols based on arithmetic variants of the LPN assumption [BCGI18, BCG+19a, WYKW20], with communication almost independent of n . VOLE on random inputs gives us a committed proof string of *random* elements; the prover can then take any of these random values and adjust them with a masked value to commit to an input of his choice.

Disjunctive proofs for IPs with LOVE (Sect. 3). Our main technical contribution, described in Sect. 3.1, can be seen as a general form of OR composition for IPs with LOVE. The communication complexity in the resulting OR proof is proportional to the *maximum* of that in the original proofs. Note that our transformation is different to the stacked garbling approach [HK20b] (which does not fit the IP with LOVE paradigm), and we obtain much greater efficiency when using our IPs with LOVE instead.

We limit ourselves to IPs with LOVE that are *public coin*, i.e., where V only sends messages that are random bits and where the queries to \mathcal{O} can be derived deterministically from the protocol transcript. This is indeed the case for our general IP with LOVE protocols that we describe later. We then go on to show that if one has m such public coin IPs with LOVE Π_1, \dots, Π_m whose messages from P to V can be made “compatible”, then one can construct a (public coin) IP with LOVE Π whose message complexity essentially only depends on the protocol Π_i which sends the most messages, plus an additional check that requires $O(m)$ communication but is independent of all m IPs of LOVE themselves. V accepts in Π if and only if at least one of the instances Π_i was accepting.

In order to run Π , P and V initially execute Π_1, \dots, Π_m in parallel. The key insight is that we only send those messages from P to V that belong to

the one protocol Π_{i^*} where P has a witness w_{i^*} for x_{i^*} , padding with dummy messages such that the communication looks as if it could belong to any of the m branches. V uses the one message that it obtains per round for all Π_1, \dots, Π_m in parallel, not knowing to which of the m protocols it belongs. Finally, instead of performing the queries to \mathcal{O} at the end of each Π_i , Π runs a standard (small) OR-proof à la Cramer et al. [CDS94] to show that the queries in at least *one* of the m branches are all valid. The trick here is that we can show that this OR-proof can itself be expressed as sending certain messages between P and V followed by queries to \mathcal{O} from V , making Π a public coin IP with LOVE as desired.

Thresholds, logarithmic overhead, and recursive nesting. The OR-proof of Cramer et al. [CDS94] can be generalized for any threshold r out of m , showing that at least r instances of Π_1, \dots, Π_m were correct. We generalize our protocol to this setting, with communication r times that of Π_i , instead of m .

While the above techniques avoid the factor m blowup from [CDS94], they do still incur an *additive* $O(m)$ overhead in the number of statements. We present a different approach, which reduces this to *logarithmic* using recursion. The key idea is that we can build a 1-out-of-2 disjunctive proof, which itself satisfies the conditions required to be stacked. Applying recursion in a binary tree-like manner, we obtain a 1-out-of- m proof with $O(\log m)$ overhead. Note that the ability to recurse is also useful when capturing proofs about complex programs, which may contain arbitrary nested levels of disjunctions, with communication proportional to the longest path through the entire program. The original stacked garbling approach [HK20b] did not support nested disjunctions, however, a later update shows how to handle them [HK20a].

Efficient IPs with LOVE for circuit satisfiability (Sect. 4). Towards efficiently instantiating IPs with LOVE, in Sect. 4.1 we describe a simple high-level syntax for expressing a large class of IPs with LOVE using an abstract homomorphic commitment notation. We refer to these as *commit-and-prove (C&P) IPs with LOVE*. This avoids the low-level details in the definition, simplifying the process of specifying and analyzing protocols. To illustrate this, we describe in Sect. 4.2 a simple protocol for circuit satisfiability.

Next, in Sect. 4.3 we present an optimized circuit satisfiability protocol that batch-checks n multiplication gates simultaneously. To achieve this, we adapt the $\log(n)$ -round inner product check of Boneh et al. [BBC+19] to C&P IPs with LOVE, which we then use for the batch check.

Due to the additive overhead generated from the batch check, it might not be the most communication-efficient approach for binary circuits when n is small. In Sect. 4.4 we therefore present a batch check of multiplications for binary circuits that has an overhead of 9 bits, essentially independent of n . This check uses *reverse multiplication-friendly embeddings* [BMN18, CCXY18] which were previously mainly used for efficient multiplications in MPC protocols.

Streaming and removing interaction (Sect. 5). We wish to obtain a zero-knowledge proof that both has a *small memory footprint*, allowing streaming, and also *minimizes interaction*, so that ideally the proof is completely non-interactive after a one-time preprocessing phase (for generating the random VOLEs). We show how to achieve a small memory footprint in our protocols by verifying each linear oracle query as it arises during the computation, rather than batching them together at the end. However, this introduces a high degree of interaction, since now the parties have to interact for every multiplication gate in the circuit.

The natural approach to avoiding interaction is to apply the Fiat-Shamir transform by obtaining the verifier's random challenges from a random oracle. However, the low-memory protocol to which we want to apply this has a very large round complexity, possibly even *linear in the circuit size*. The Fiat-Shamir transform is typically only applied to constant-round protocols, since in the worst-case, the soundness can degrade *exponentially* with the number of rounds [BCS16]. Several works, however, have defined extra conditions on the underlying protocol which suffice to avoid this degradation, for the cases of interactive oracle proofs [BCS16] and general interactive proofs [CCH+19].

Following in this direction, we adapt the concept of *round-by-round soundness* [CCH+19] of interactive proofs to IPs with LOVE. We then show that by applying a Fiat-Shamir transform, any IP with LOVE satisfying this modified notion can be transformed into a NIZK (with VOLE preprocessing) in the random oracle model, with negligible soundness degradation. Finally, we also show that our streamable protocols for circuit satisfiability do indeed have round-by-round soundness, so can safely be made non-interactive.

2 Interactive Proofs with Linear Oracle Verification

In this section we introduce our proof methodology, called *interactive proofs with linear oracle verification* (IPs with LOVE). In addition, we show how, using vector oblivious linear evaluation (VOLE), any public coin IP with LOVE can be turned into a zero-knowledge proof.

Notation. For any vector \mathbf{r} we denote by $\mathbf{r}|_t$ the restriction to the first t elements and by $\mathbf{r}[i]$ the i th element of \mathbf{r} . Let $[\mathbf{P} \leftrightarrow \mathbf{V}]$ denote the distribution of exchanged messages between two parties \mathbf{P} and \mathbf{V} and let $[\mathbf{P} \leftrightarrow \mathbf{V}]_t$ denote the distribution of the transcript of the messages exchanged in the first t rounds. Denote by $\text{View}_{\mathbf{V}}[\mathbf{P} \leftrightarrow \mathbf{V}]$ the view of \mathbf{V} when interacting with \mathbf{P} . We defer the (standard) definition of zero-knowledge proofs to the full version.

2.1 Definitions

We now formalize IPs with LOVE over a finite field \mathbb{F}_{p^k} . This formalization is a generalization of linear interactive oracle proofs [BBC+19], where in each round, the verifier chooses some linear function, and learns the evaluation of this on a proof string chosen by the prover. In comparison, we let the prover \mathbf{P} first fix the proof string $\boldsymbol{\pi}$, which is a vector of field elements. Then, both \mathbf{P} and the

verifier V exchange messages for a certain number of rounds. Finally, V issues a number of affine queries to π , upon which it makes a decision on whether to accept or not. These queries can depend on the messages that were exchanged between both P and V throughout the protocol.

We let P fix $\pi \in \mathbb{F}_{p^k}^\ell$ at the beginning of the protocol and allow V to access it via oracle queries only at the end of the protocol. In the oracle query stage, we let V choose q queries $(z_1, y_1), \dots, (z_q, y_q) \in \mathbb{F}_{p^k}^\ell \times \mathbb{F}_{p^k}$ which it sends to an oracle that stores π . This oracle checks that for each of the q queries the relation $\langle \pi, z_i \rangle = y_i$ holds. The query results are then (truthfully) reported to V by the oracle.

Note that by default, both π and the queries lie over the extension field \mathbb{F}_{p^k} . In some cases, such as when we are proving statements over \mathbb{F}_p , some elements of π may only be in \mathbb{F}_p , which allows for improved efficiency when instantiating IPs with LOVE, as we will see later. In this case, during the query phase we view any \mathbb{F}_p value as an element of \mathbb{F}_{p^k} via some fixed embedding.

Definition 1 (Interactive Protocol with Linear Oracle Verification).

Let \mathbb{F}_{p^k} be a field and $\ell, t, q \in \mathbb{N}$. Then a t -round q -query interactive protocol with linear oracle verification $\Pi = (P, V)$ with oracle length ℓ , message lengths $r_1^P, r_1^V, \dots, r_t^P, r_t^V \in \mathbb{N}$ and message complexity $\sum_{h=1}^t (r_h^P + r_h^V)$ over \mathbb{F}_{p^k} consists of the algorithm P and PPT algorithm V that interact as follows:

1. Initially, P obtains its respective input while V obtains the statement x . P then submits a string $\pi \in \mathbb{F}_{p^k}^\ell$ to the oracle. P then outputs a state s_0^P while V outputs a state s_0^V . We set an auxiliary variable $a_0 = \perp$.
2. For round $h \in [t]$, P and V do the following:
 - (a) First, V on input s_{h-1}^V and a_{h-1} outputs message $e_h \in \mathbb{F}_p^{r_h^V}$ and state s_h^V .
 - (b) Then, P on input s_{h-1}^P and e_h outputs message $a_h \in \mathbb{F}_p^{r_h^P}$ and state s_h^P .
3. Finally, V on input a_t and state s_t^V makes q linear oracle queries to π over \mathbb{F}_{p^k} and outputs a bit.

We say that the protocol accepts if V outputs 1 at the end of the protocol.

Remark 1. Note that the prover and verifier's messages (a_h, e_h) are specified as elements of the base field \mathbb{F}_p , and this is how we count message complexity. This is an arbitrary restriction, since these messages can easily be used to encode extension field elements or general bit strings.

Definition 2 (Honest-Verifier Zero-Knowledge Interactive Proof with Linear Oracle Verification).

A t -round q -query interactive protocol with linear oracle verification $\Pi = (P, V)$ over \mathbb{F}_{p^k} is an honest-verifier zero-knowledge interactive proof with linear oracle verification (HVZK IP with LOVE) for a relation \mathcal{R} with soundness error ϵ if it satisfies the following three properties:

Completeness: For all $(x, w) \in \mathcal{R}$ the interaction between $P(x, w)$ and $V(x)$ is accepting.

Soundness: For all $x \notin L(\mathcal{R})$ and for all (unbounded) algorithms P^* , any interaction of P^* with $\mathsf{V}(x)$ is accepting with probability at most ϵ .

Honest-Verifier Zero-Knowledge: There exists a PPT algorithm S such that for any $(x, \mathbf{w}) \in R$ the output of $S(x)$ is perfectly indistinguishable from $\text{View}_{\mathsf{V}}[\mathsf{P}(x, \mathbf{w}) \leftrightarrow \mathsf{V}(x)]$ for any honest V .

We use the notation IP-LOVe to denote a t -round, q -query HVZK IP with LOVe for relation \mathcal{R} over field \mathbb{F}_{p^k} with oracle length ℓ , message complexity α elements of \mathbb{F}_p , and soundness error ϵ .

In this work, all the protocols we construct will additionally be proofs of knowledge and public coin, as in the following definitions.

Definition 3 (ZK Interactive Proof of Knowledge with LOVe). Let Π be an IP-LOVe protocol for a statement x using a proof string π such that V accepts with probability $> \epsilon$. Then Π is a proof of knowledge if there exists a PPT extractor E that, on input x, π , outputs a witness \mathbf{w} such that $(x, \mathbf{w}) \in \mathcal{R}$.

Definition 4 (Public Coin IP with LOVe). An IP-LOVe protocol Π is public coin if

1. V chooses each $e_h \in \mathbb{F}_p^{n_h^{\mathsf{V}}}$ for $x \in L(\mathcal{R})$ uniformly at random (and in particular, independent of s_{h-1}^{V} and \mathbf{a}_{h-1}).
2. There exists a deterministic polytime algorithm \mathcal{Q} , which, on input x and $\{e_h, \mathbf{a}_h\}_{h \in [t]}$, computes the q oracle queries $(z_1, y_1), \dots, (z_q, y_q)$ of V .
3. V accepts iff all queries generated by \mathcal{Q} are accepting.

From q -query to 1-query. Given a q -query IP with LOVe, we can always convert it to one with a single oracle query, with a small loss in soundness, by taking random linear combinations of all queries over a large enough extension field. This transformation, given below, is public-coin and adds just one extra round of communication, so when using IPs with LOVe, we will often assume they have only one query, to simplify our protocols.

Let Π be an IP-LOVe over \mathbb{F}_p (p need not be prime), and let k be such that p^k is superpolynomial in a statistical security parameter. We construct an IP with LOVe over \mathbb{F}_{p^k} , by viewing the proof $\pi \in \mathbb{F}_p^\ell$ from Π as a vector in $\mathbb{F}_{p^k}^\ell$, running the same protocol and then modifying the query phase as follows. Recall that the q queries $(z_1, y_1), \dots, (z_q, y_q)$ in Π accept if and only if $\langle \pi, z_i \rangle = y_i$ i.e. $\mu_i := \langle \pi, z_i \rangle - y_i = 0$ in \mathbb{F}_p . Now, we modify the protocol by having V send q random elements $\rho_1, \dots, \rho_q \in \mathbb{F}_{p^k}$ to P .² Notice that if $\mu := \sum_{i \in [q]} \rho_i \mu_i = 0$, all queries are satisfied except with probability p^{-k} . We equivalently have $\mu = \langle \pi, \mathbf{z} \rangle - y$ for $\mathbf{z} = \sum_{i \in [q]} \rho_i z_i$ and $y = \sum_{i \in [q]} \rho_i y_i$. This shows we can reduce the q oracle queries down to just one query (\mathbf{z}, y) over \mathbb{F}_{p^k} , at the cost of an extra q elements of \mathbb{F}_{p^k} sent from V to P , and the soundness error increasing by p^{-k} .³

² If we did not want a public-coin protocol, we could skip this message from V to P .

³ Alternatively, V could send a single random $\rho \in \mathbb{F}_{p^k}$, and define $\rho_i = \rho^i$. This reduces communication while increasing the error probability to $q \cdot p^{-k}$, by applying the Schwartz-Zippel Lemma.

2.2 Instantiating IPs with LOVE Using VOLE

We now show that any IP-LOVe can be transformed into a zero-knowledge proof, by using *vector oblivious linear evaluation (VOLE)* to instantiate the linear oracle queries. The functionality for random VOLE is given in Fig. 1: it picks a vector of random samples $(\mathbf{r}, \boldsymbol{\tau}), (\alpha, \boldsymbol{\beta})$ such that $\boldsymbol{\tau} = \mathbf{r}\alpha + \boldsymbol{\beta}$, and outputs them to the respective parties. This can be seen as a secret-sharing of the products $\mathbf{r}[i]\alpha$, for $i = 1, \dots, \ell$. Note that we relax security slightly by allowing corrupt parties to choose their own randomness. This models existing random VOLE protocols based on the LPN assumption [BCGI18, BCG+19b, WYKW20], which can generate a large, length ℓ VOLE with communication that is almost independent of ℓ .

Commitments with MACs. We can view each output of a VOLE as an information-theoretic MAC on the value $\mathbf{r}[i]$, which commits the prover to $\mathbf{r}[i]$. We write $[x]$ to denote that the prover holds $x, \tau_x \in \mathbb{F}_{p^k}$, while the verifier holds β_x and the fixed MAC key $\alpha \in \mathbb{F}_{p^k}$. To open a commitment to x , the prover sends x, τ_x and the verifier checks that $\tau_x = x\alpha + \beta_x$. It is easy to see that cheating in an opening requires guessing the random MAC key α , so happens with probability $1/p^k$.

Since α is the same for each commitment, these commitments are *linearly homomorphic*. Indeed, given two commitments $[x], [y]$, the parties can obtain $[x + y]$ by computing $x + y, \tau_x + \tau_y$ and $\beta_x + \beta_y$, respectively. Similarly, we can do multiplication by constant, and addition by constant c (here, the verifier adds αc to β_x , while the prover adds c to x). We overload the $+$ and \cdot operators to denote these operations being performed on the commitments.

The transformation (Fig. 2). Given the linearly homomorphic commitment scheme based on VOLE, obtaining a ZK proof is relatively straightforward. First, the prover commits to its proof string $\boldsymbol{\pi}$, by sending each component masked with a random VOLE commitment. The parties then run the IP-LOVe protocol as usual, until the query phase. Here, each linear query is computed by applying the linear function to the committed $\boldsymbol{\pi}$, followed by opening the result to check it gives the correct value. In the full version, we prove the following.

Theorem 1. *Suppose Π_{LOVe} is a public-coin IP-LOVe for relation \mathcal{R} , satisfying completeness, soundness error ϵ and honest-verifier zero-knowledge. Then, $\Pi_{\text{ZK}}^{\text{VOLE}}$ is an honest-verifier zero-knowledge proof for relation \mathcal{R} , with soundness error $\epsilon + p^{-k}$. Furthermore, if Π_{LOVe} is a proof of knowledge, then so is $\Pi_{\text{ZK}}^{\text{VOLE}}$.*

Optimizations: Random proof elements, and subfield VOLE. We describe two simple optimizations, which reduce communication in certain cases.

Firstly, in our protocols, the proof string $\boldsymbol{\pi}$ will often contain many random field elements; clearly, the values d_i in Step 2 of the Input Phase (cf. Fig. 2) do not need to be sent in this case, since P can choose $\boldsymbol{\pi}[i] = \mathbf{r}[i]$.

Secondly, when working over an extension field \mathbb{F}_{p^k} , sometimes it is known that $\boldsymbol{\pi}$ will consist mainly of elements in the base field \mathbb{F}_p (viewed as a subset

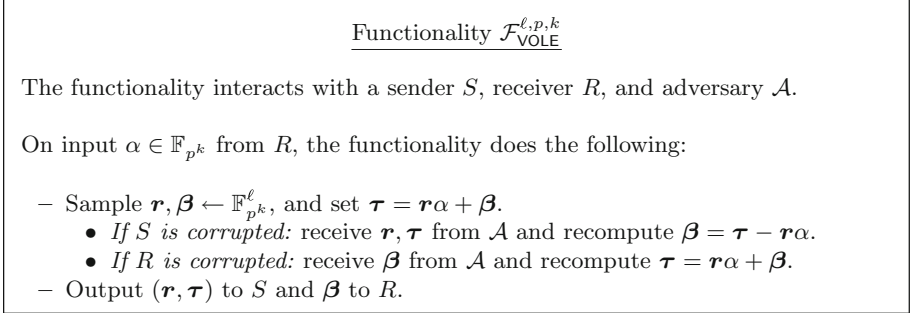


Fig. 1. Ideal functionality for vector oblivious linear evaluation over \mathbb{F}_{p^k} .

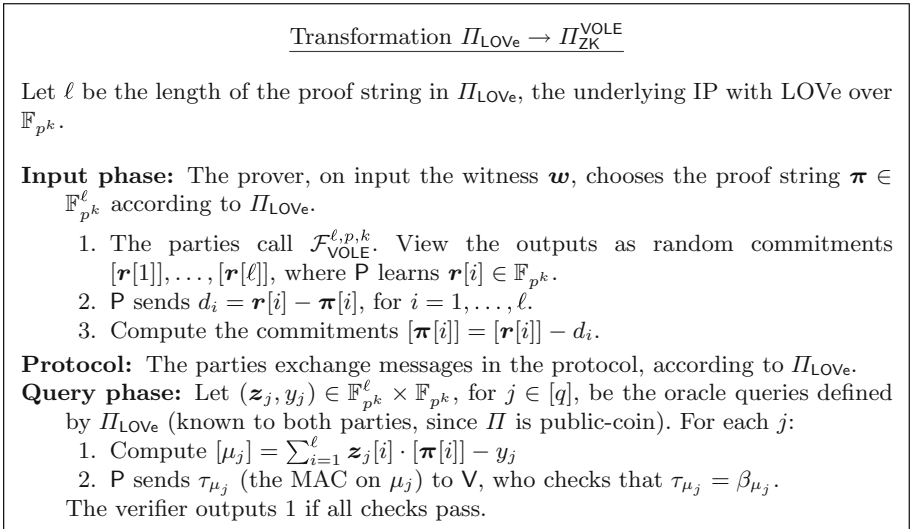


Fig. 2. Zero-knowledge proof from VOLE and IP with LOVe.

of \mathbb{F}_{p^k} by a fixed embedding). In this case, we can optimize communication by using *subfield* VOLE instead of VOLE over \mathbb{F}_{p^k} . In subfield VOLE [BCG+19b], \mathbf{r} is sampled as a uniform vector over \mathbb{F}_p instead of \mathbb{F}_{p^k} , while the MACs $\boldsymbol{\tau}$ and keys α, β are still computed over \mathbb{F}_{p^k} . This allows P to commit to values from the subfield \mathbb{F}_p (which may be small), by sending only elements from \mathbb{F}_p , while still achieving soundness error p^{-k} . Note that this still allows committing to an extension field element $x \in \mathbb{F}_{p^k}$ if needed, by decomposing x into a linear combination of \mathbb{F}_p elements, and committing to each component separately (this works because the MACs are linear over \mathbb{F}_{p^k}).

When analyzing the complexity of our protocols, we assume that the above two optimizations have been applied.

3 Stackable Public Coin IPs with LOVE

In this section, we show that when both P and V agree on m relations $\mathcal{R}_1, \dots, \mathcal{R}_m$ and instances x_1, \dots, x_m that can each be proven using (public coin HVZK) IPs with LOVE over the same field \mathbb{F}_p , then we can construct a communication-efficient protocol showing that at least one of the statements was true. Following the terminology of stacked garbling [HK20b], we sometimes refer to this as a *stacked proof*. Formally, the goal of P is to show that $(x_1, \dots, x_m) \in L(\mathcal{R}_{\text{OR}})$ where

$$(x_1, \dots, x_m) \in L(\mathcal{R}_{\text{OR}}) \iff x_1 \in L(\mathcal{R}_1) \vee \dots \vee x_m \in L(\mathcal{R}_m).$$

Throughout this section, we will write \hat{x} as a short-hand for x_1, \dots, x_m when the statements are clear from the context. Suppose we have IPs with LOVE over \mathbb{F}_p for each instance x_i , with message complexity α_i . The classic OR-proof technique by Cramer et al. [CDS94] can be used to give an IP with LOVE with message complexity $\approx \sum_{i \in [m]} \alpha_i$. This would be done by running all m proofs in parallel (which means sending messages for all of them), and then showing that at least one finished with the expected output using [CDS94]. We show how to instead reduce the message complexity of such a proof to $2mk + \max\{\alpha_i\}$, where the soundness error grows by $\approx p^{-k}$. We also give a variant where the message complexity scales with $O(\log m)$, instead of $2m$.

Towards this, we introduce the notion of equisimulatable IPs with LOVE. The idea is that we can compress the messages for the different proof branches sent by P in such a way that for the true branch, the correct message can be recovered by V . The distribution of the values for non-taken branches which V will obtain is indistinguishable from a real protocol execution.

For example, assume that in the Π_1 branch, P sends one \mathbb{F}_p element that appears uniformly random to V , while in the Π_2 branch it sends two such elements with the same property. To achieve equisimulatability, if P actually proves the first branch to be true then it can always append a uniformly random element to the message it sends to V , whereas in the second case it just sends the actual message. In both cases, the distribution of the message sent by P is identical and V cannot identify which branch was taken by P .

Formally, for m statements with protocols Π_1, \dots, Π_m , we use the following two algorithms, which should satisfy the definition below.

- The “combined prover” algorithm \mathcal{CP} takes as input instances x_1, \dots, x_m , instance index i , round index h , and prover message $\mathbf{a}_h \in \mathbb{F}_p^{\mathcal{R}_i}$, and outputs a message $c \in \mathbb{F}_p^*$, which encodes \mathbf{a}_h while disguising it to hide the index i .
- The “decode” algorithm dec takes as input instances x_1, \dots, x_m , index i , round index h , and combined prover message $c \in \mathbb{F}_p^*$, and recovers $\mathbf{a}'_h \in \mathbb{F}_p^{\mathcal{R}_i}$.

Definition 5 (Equisimulatable IPs with LOVE). *Let Π_1, \dots, Π_m be protocols such that each Π_i is an IP with LOVE over \mathbb{F}_p for the relation \mathcal{R}_i with round complexity t_i . We say that Π_1, \dots, Π_m are equisimulatable if there exist two algorithms \mathcal{CP} and dec such that:*

1. If $\mathbf{a}_h \leftarrow \Pi_i(s_{h-1}^P, \mathbf{e}_h)$, where Π_i 's inputs are from an honest execution of Π_i , then

$$\text{dec}(\hat{x}, i, h, \mathcal{CP}(\hat{x}, i, h, \mathbf{a}_h)) = \mathbf{a}_h.$$

2. For any i, j , the distributions $\{\mathcal{CP}(\hat{x}, i, h, \mathbf{a}_h) \mid \mathbf{a}_h \leftarrow \Pi_i(s_{h-1}^P, \mathbf{e}_h)\}_{h \in [t_i]}$ and $\{\mathcal{CP}(\hat{x}, j, h, \mathbf{a}_h) \mid \mathbf{a}_h \leftarrow \Pi_j(s_{h-1}^P, \mathbf{e}_h)\}_{h \in [t_j]}$, where both the inputs of Π_i and Π_j come from honest executions, are perfectly indistinguishable.

We say that \mathcal{CP} has message complexity α if the total number of \mathbb{F}_p elements generated by \mathcal{CP} for all $h \in [\max_{i \in [m]} t_i]$ is at most α .

In our constructions of IPs with LOVE, all prover messages will appear uniformly random. We show below that this implies both the zero-knowledge property and equisimulatability, which gives us an easy criterion for proving that an IP-LOVe can be stacked. We prove the following lemma in the full version.

Lemma 1. *Let Π be an IP-LOVe for proving relation \mathcal{R} , satisfying completeness, where \mathbf{V} accepts iff all queries are accepting. If the messages from \mathbf{P} in an honest execution are (perfectly) indistinguishable from random, it holds that*

1. Π is honest-verifier zero-knowledge; and
2. m such instances of Π (potentially for different relations $\mathcal{R}' \neq \mathcal{R}$) are equisimulatable (cf. Definition 5).

3.1 Stacking with LOVE

Using the concept of *equisimulatability* of protocols we now show how to lower the message complexity when proving \mathcal{R}_{OR} . The protocol, given in Fig. 3, is inspired by the stacked garbling approach [HK20b], although uses a very different technique.

We start with m equisimulatable IPs with LOVE Π_i , over \mathbb{F}_p , for proving individual relations \mathcal{R}_i . Note that p can be any prime power, with no restrictions on size. We construct a protocol Π_{OR} , defined over \mathbb{F}_{p^k} , which works as follows. \mathbf{P} , having only w_{i^*} for one of the statements x_{i^*} , will generate the oracle string π by running Π_{i^*} 's first step to create π_{i^*} , which it then pads with extra random data, and embeds in to \mathbb{F}_{p^k} . Then, \mathbf{P} and \mathbf{V} will *simultaneously* run all Π_1, \dots, Π_m , with the following modification: \mathbf{P} 's message c_h to \mathbf{V} in round h will be determined from $\mathbf{a}_{h,i}$ using the combined prover algorithm \mathcal{CP} , while \mathbf{V} extracts the message $\mathbf{a}_{h,i}$ for each of the instances from c_h using dec . Due to equisimulatability, \mathbf{V} can now execute all instances in parallel but cannot tell which of these is the true one. Conversely, since all Π_i are public coin, \mathbf{V} sends a randomness string that is long enough for any of the m instances in round h . The message complexity is now determined by \mathcal{CP} and not the individual proofs.

The challenge now, is that \mathbf{V} cannot simply perform the oracle queries for all Π_i , since this would reveal the index i^* of the true statement. Instead, we perform a [CDS94]-style OR-proof to show that at least one of the query's for the Π_i is accepting. Recall, the basic idea behind [CDS94] is that given m Σ -protocols for proving relations, an OR proof can be done by having the *prover*

choose the random challenge f_i for $m - 1$ of the instances, so it can simulate the correct messages to be sent in every false instance, without knowing a witness. Then, after receiving the m initial messages of each Σ -protocol, the verifier picks a challenge f , which defines the challenge $f_{i^*} = f - \sum_i f_i$ corresponding to the true instance i^* (while hiding i^* from V).

We instantiate the above, where each “instance” corresponds to a small protocol for verifying that the oracle query $\langle \pi_i, z_i \rangle = y_i$ for protocol Π_i succeeds, without actually performing the query. To carry out one such small protocol, P includes an extra random value r_i in the proof string $\pi := (\pi_i || r_i)$ (one such r_i for each branch).

For the true Π_{i^*} the prover later sends $d_{i^*} := r_{i^*}$ to V . Using the random challenge f_{i^*} , the verifier then makes a query to test that $\langle \pi, z_{i^*} || f_{i^*} \rangle =? y_{i^*} + f_{i^*} d_{i^*}$. This clearly accepts if $d_{i^*} = r_{i^*}$ and the original query $(y_{i^*} =? \langle \pi_{i^*}, z_{i^*} \rangle)$ accepts.

Importantly, if P does *not* know a valid witness, but can pick f_i in advance, then P can cheat by setting $d_i = (\langle \pi_i, z_i \rangle - y_i) / f_i + r_i$, causing the aforementioned oracle query to succeed as well. This is the crux of Phase II of the protocol in Fig. 3.

Note that the theorem below assumes that each protocol Π_i uses only one query. As discussed at the end of Sect. 2.1, this can always be achieved by combining queries into one (at the cost of one additional round). The proof of the following theorem can be found in the full version.

Theorem 2. *Let Π_1, \dots, Π_m be protocols such that each Π_i is a t_i -round, 1-query, equisimulatable Public Coin IP with LOVE over \mathbb{F}_p for relation \mathcal{R}_i with oracle length ℓ_i and soundness error ϵ_i . Furthermore, assume that \mathcal{CP} has overall message complexity α . Then the protocol Π_{OR} in Fig. 3 is a Public Coin IP with LOVE over \mathbb{F}_{p^k} for the relation \mathcal{R}_{OR} with*

1. round complexity $3 + \max_{i \in [m]} t_i$;
2. oracle length $m + \max_{i \in [m]} \ell_i$;
3. query complexity m ;
4. message complexity $2mrk + \alpha$ elements of \mathbb{F}_p ; and
5. soundness error $\sum_{i \in [m]} \epsilon_i + 1/p^k$.

If Π_1, \dots, Π_m are all proofs of knowledge, then so is Π_{OR} .

Generalizing to threshold proofs. In [CDS94] the authors describe how to additionally construct proofs of partial knowledge for any threshold, i.e., how to show that r out of the m statements are true. Their technique, together with a modification of Π_{OR} , can be used to construct a proof in our setting where we implicitly only communicate the transcript of r statements, and not all m of them. Π_{OR} can then be seen as a special case where $r = 1$, where for general r we use Shamir secret-sharing, instead of additive shares of the verifier’s challenge f . More details are found in the full version.

Protocol Π_{OR}

Let Π_1, \dots, Π_m be protocols such that each Π_i is t_i -round, 1-query equisimulatable public coin IP with LOVe over \mathbb{F}_p for relation \mathcal{R}_i with oracle length ℓ_i .

Both P and V have inputs x_1, \dots, x_m where $x_i \in L(\mathcal{R}_i)$. P additionally has input \mathbf{w}_{i^*} for (at least) one $i^* \in [m]$ such that $(x_{i^*}, \mathbf{w}_{i^*}) \in \mathcal{R}_{i^*}$. We define $\ell := \max_{i \in [m]} \ell_i$, and $t := \max_{i \in [m]} t_i$. Let $\mathbf{z}_{k,i} = \overline{\mathbf{z}}_{k,i} \underbrace{\| 0 \cdots 0}_{\ell - \ell_i \text{ times}}$.

1. P simulates Π_{i^*} on input $(x_{i^*}, \mathbf{w}_{i^*})$ to obtain the string $\boldsymbol{\pi}_{i^*}$. It then sets

$$\boldsymbol{\pi}' = \boldsymbol{\pi}_{i^*} \|\underbrace{0 \cdots 0}_{\ell - \ell_{i^*} \text{ times}} \quad \text{and} \quad \boldsymbol{\pi} = \boldsymbol{\pi}' \| r_1 \cdots r_m$$

where all r_i are chosen uniformly at random in \mathbb{F}_{p^k} .

(Phase I: Running the stacked proof)

2. Define $s_0^{\text{P}} := (x_{i^*}, \mathbf{w}_{i^*})$. For $h \in [t]$, P and V do the following:
 - (a) Let $r_{h,i}^{\text{V}}$ be the length of the challenge that V would send for protocol Π_i in round h . V sets $r_h = \max_{i \in [m]} r_{h,i}^{\text{V}}$, samples $\mathbf{e}_h \leftarrow \mathbb{F}_p^{r_h}$ uniformly at random and then sends it to P.
 - (b) P sets $(\mathbf{a}_h, s_h^{\text{P}}) \leftarrow \Pi_{i^*}(s_{h-1}^{\text{P}}, \mathbf{e}_h)$ where P only uses the first r_{h,i^*}^{P} elements of \mathbf{e}_h as required by Π_{i^*} . It then computes $c_h \leftarrow \mathcal{CP}(\hat{x}, h, i^*, \mathbf{a}_h)$ and sends c_h to V.

(Phase II: Running the small OR proof)

3. For $i \in [m] \setminus \{i^*\}$, P samples $f_i \leftarrow \mathbb{F}_{p^k}^*$ uniformly at random and computes $(\mathbf{z}_i, y_i) \leftarrow \mathcal{Q}(x_i, \{\mathbf{e}_h, \text{dec}(\hat{x}, h, i, c_h)\}_{h \in [t_i]})$. It then computes $d_i := (\langle \boldsymbol{\pi}', \mathbf{z}_i \rangle - y_i) / f_i + r_i$, and defines $d_{i^*} := r_{i^*}$. Finally, P sends $(d_1, \dots, d_m) \in \mathbb{F}_{p^k}^m$ to V.
4. V samples $f \leftarrow \mathbb{F}_{p^k}$ uniformly at random and sends it to P.
5. P sets $f_{i^*} := f - \sum_{i \in [m] \setminus \{i^*\}} f_i$ and sends f_1, \dots, f_{m-1} to V. V computes the last challenge $f_m = f - \sum_{i=1}^{m-1} f_i$.
6. Let $\boldsymbol{\beta}_i \in \mathbb{F}_{p^k}^m$ be the vector that is f_i in the i th position and 0 everywhere else. For $i \in [m]$, V first generates (\mathbf{z}_i, y_i) like P in Step 3. Then, for each $i \in [m]$ it sends the query $(\mathbf{z}_i \| \boldsymbol{\beta}_i, y_i + f_i d_i)$ to the oracle. V accepts if all queries are true.

Fig. 3. The protocol Π_{OR} for an OR-statement.

3.2 Recursive Stacking

Π_{OR} from Sect. 3.1 has the drawback that to verify one out of m statements, we still need $O(m)$ communication complexity. We now give an alternative construction that obtains an overhead only *logarithmic in m* .

The idea behind this alternative protocol is as follows:

1. Any IP-LOVe Π accepts iff all queries are accepting. Assuming (wlog) there is only one query, this means that for the query (z, y) , we have $\langle \boldsymbol{\pi}, z \rangle = y$ i.e. $\mu := \langle \boldsymbol{\pi}, z \rangle - y = 0$.

2. If we simulate the parallel evaluation of m protocol instances as in Π_{OR} , then if for any branch i^* it holds that $\mu_{i^*} = 0$, then i^* must correspond to a “true” branch.
3. If the prover can then compute the product $\mu_1 \cdots \mu_m$, and prove that this is 0, then at least one μ_j was 0 to begin with.

A naive instantiation of the above approach is to perform $m - 1$ multiplications between the m implicit variables μ_i , and open the result. However, this would still give $O(m)$ overhead. Instead, we carefully apply recursion to make this overhead logarithmic. Here, we use the fact that after combining two protocols Π_1, Π_2 with the multiplication method sketched above, we can obtain a protocol which *itself is again stackable*: considering all multiplications as a tree, we only have to provide those values necessary to prove a correct multiplication that are on the path from μ_{i^*} to the root.

The actual proof for this proceeds in the following steps:

1. First we show that if Π_1, Π_2 fulfill similar conditions as in Π_{OR} then we can combine them using the multiplication-based approach.
2. Next, we show that starting with $2m$ proofs Π_1, \dots, Π_{2m} with similar conditions as in Π_{OR} , if we construct proofs Π'_i from Π_{2i-1}, Π_{2i} using the multiplication method, then Π'_1, \dots, Π'_m again fulfill the same conditions i.e. are stackable. Also, this can be done with an overhead that is only as big as one Π_i plus one multiplication.
3. Finally, by recursing the previous step, we obtain the log-overhead OR-proof.

The full construction, together with its proof, can be found in the full version. One drawback of this approach, though, is that unlike our previous OR-proof based method, it does not give rise to a t -out-of- m proof.

4 IPs with LOVE for Circuit Satisfiability

In this section, we present our protocols for proving circuit satisfiability of arithmetic and boolean circuits. First, in Sect. 4.1, we define a high-level *commit-and-prove* (C&P) syntax for IPs with LOVE. This makes it simpler to specify protocols, and also aligns with the VOLE instantiation used in Sect. 2.2. We then describe a simple protocol for arithmetic circuit satisfiability over a finite field \mathbb{F}_p (Sect. 4.2), with communication cost of 3 field elements per multiplication gate for large p . We next show how we can utilize fully linear PCPs by Boneh et al. [BBC+19] to reduce the amortized multiplication cost to just over 1 \mathbb{F}_p element per multiplication gate (Sect. 4.3), when the circuit size is large enough. The same approach also works over *binary* fields with the same cost (Sect. 4.4).

To highlight the power of our disjunctive proof from Sect. 3.1, we point out that all of these protocols fulfil the criteria of our stacking approach, so lead to efficient proofs of disjunctions. Recall that from Lemma 1, it suffices that the sender’s messages in the IP-LOVe are uniformly random, which we show for all protocols in this section.

4.1 Defining C&P Protocols

We now define a high-level, commit-and-prove (C&P) syntax for specifying a large class of IPs with LOVE over \mathbb{F}_{p^k} .

We require that the witness in the IP with LOVE is a vector $\mathbf{w} = (w_1, \dots, w_n)$ of \mathbb{F}_p elements, and that the prover chooses the proof string $\pi = (w_1, \dots, w_n, r_1, \dots, r_t)$, where each r_i is uniformly random. As remarked in Sect. 2.2, we may sometimes wish to mix values in \mathbb{F}_p and \mathbb{F}_{p^k} , so allow the possibility that some r_i 's are sampled from \mathbb{F}_{p^k} and others are in the base field.

Following the notation used for homomorphic commitments in Sect. 2.2, we write $[x]$ to denote that some value x is committed to by the prover P. Initially, P is committed to every element w_i, r_i of the proof string π . Subsequently, we allow the parties to perform affine operations on these committed values, obtaining new commitments.

Finally, we model the linear verification oracle by a special instruction `AssertZero`, which checks whether its input is a commitment to 0. Since any commitment comes from an affine function of π , this exactly models linear queries to π . We then specify a C&P protocol over \mathbb{F}_{p^k} as follows:

Input phase: P has input the witness $w_1, \dots, w_n \in \mathbb{F}_p$, and samples random values $r_1, \dots, r_t \leftarrow \mathbb{F}_{p^k}^t$ (optionally, some r_i 's may be in \mathbb{F}_p).

P inputs the proof string $\pi = (w_1, \dots, w_n, r_1, \dots, r_t)$.

Protocol phase: The parties, given commitments $[w_1], \dots, [w_n]$, run a sequence of instructions of the following types:

- `Random(\mathbb{F})` (for $\mathbb{F} \in \{\mathbb{F}_p, \mathbb{F}_{p^k}\}$): Retrieve $[r]$, where $r \in \mathbb{F}$ is the next suitable random value in π .
- `Send $_{[P \rightarrow V]}(x)$` : Sends value $x \in \mathbb{F}_p$ from P to V.
- `Send $_{[V \rightarrow P]}(x)$` : Sends value $x \in \mathbb{F}_p$ from V to P.
- `$[z] = a[x] + b[y] + c$` : Define the commitment $[z]$ for $z = ax + by + c$, given some public values a, b, c .
- `AssertZero($[x]$)`: Asserts to V that $[x]$ is a commitment to $x = 0$.

Output phase: If none of the `AssertZero` instructions failed, the verifier outputs 1. Otherwise, it outputs 0.

As described previously, by translating `AssertZero` calls into linear oracle queries, any C&P protocol specified in the above syntax defines a valid IP-LOVE.

4.2 C&P IP with LOVE for Arithmetic Circuits

We now show a C&P IP with LOVE for arithmetic circuit satisfiability that satisfies (1) completeness, (2) soundness, and (3) that all inputs to `Send` are indistinguishable from random. Thus, by Lemma 1 we conclude that our protocol is also zero knowledge and supports disjunctions. We prove circuit satisfiability over a field \mathbb{F}_p , but define a protocol over \mathbb{F}_{p^k} for some $k \geq 1$, so that soundness can be boosted if necessary.

We begin by defining two auxiliary “instructions”: (1) `Fix`, which allows P to fix a random commitment to a value of its choosing, and (2) `Reveal`, which opens a commitment to V and checks this was done properly using `AssertZero`.

- $\text{Fix}(x) \rightarrow [x]$: On input $x \in \mathbb{F}$ (where $\mathbb{F} \in \{\mathbb{F}_p, \mathbb{F}_{p^k}\}$) from P , output a commitment $[x]$. This is implemented as:
 1. $\text{Random}(\mathbb{F}) \rightarrow [r]$.
 2. $\text{Send}_{[\mathsf{P} \rightarrow \mathsf{V}]}(x - r) \rightarrow y$.
 3. $[r] + y \rightarrow [x]$.
- $\text{Reveal}([x]) \rightarrow x$: On input commitment $[x]$, output x to V . This is implemented as:
 1. $\text{Send}_{[\mathsf{P} \rightarrow \mathsf{V}]}(x)$.
 2. $\text{AssertZero}([x] - x)$.

Our protocol works as follows. Let $C : \mathbb{F}_p^n \rightarrow \mathbb{F}_p$ be a circuit known to both parties consisting of **Add** and **Mult** gates, which we want to show evaluates to zero on some input. The prover provides the witness $\mathbf{w} \in \mathbb{F}_p^n$ as input, so the parties initially get commitments $[w_1], \dots, [w_n]$. The parties then execute the following steps to evaluate the circuit C , where we denote by C^* the set of multiplication gates in C .

1. For each gate in C , in topological order, proceed as follows:
 - Add** $([x], [y])$: Output $[x] + [y]$.
 - Mult** $([x], [y])$: Run $\text{Random}(\mathbb{F}_{p^k}) \rightarrow [a]$, $\text{Fix}(xy) \rightarrow [z]$, and $\text{Fix}(ay) \rightarrow [c]$. Output $[z]$, and store the commitments $[a]$ and $[c]$.
2. Run $\text{Send}_{[\mathsf{V} \rightarrow \mathsf{P}]}(e)$, where $e \in_R \mathbb{F}_{p^k}$.
3. For each $i \in [|C^*|]$ let $[x_i]$ and $[y_i]$ denote the inputs and $[z_i]$, $[a_i]$ and $[c_i]$ denote the outputs and stored values in the i -th call to **Mult**. Then run $\text{AssertMult}([x_i], [y_i], [z_i], [a_i], [c_i], e)$.
4. Run $\text{AssertZero}([z_{\text{out}}])$, where $[z_{\text{out}}]$ is the commitment to the output of C .

The subprotocol **AssertMult** used above works as follows:

- AssertMult** $([x], [y], [z], [a], [c], e)$:
1. Run $\text{Reveal}([\varepsilon])$, where $[\varepsilon] = e[x] - [a]$.
 2. Run $\text{AssertZero}(e[z] - [c] - \varepsilon[y])$.

Before proving security, observe that the communication complexity is 3 field elements per multiplication gate, of which one is over \mathbb{F}_p (for fixing xy) and two over \mathbb{F}_{p^k} (for fixing ay , and revealing ε).

Theorem 3. *Let \mathcal{R} be a relation that can be represented by an arithmetic circuit C over \mathbb{F}_p such that $\mathcal{R}(x, \mathbf{w}) = 1 \Leftrightarrow C(\mathbf{w}) = 0$. Then the above protocol is a C&P IP with LOVE over \mathbb{F}_{p^k} for \mathcal{R} , such that (1) completeness holds, (2) soundness holds with soundness error p^{-k} , (3) all inputs to **Send** are perfectly indistinguishable from random, and (4) the protocol is a proof of knowledge.*

The proof can be found in the full version. At a high level, soundness holds by the security of the **Mult** operation, where a malicious prover essentially needs to align its invalid **Fix** values with the verifier's random e value, which happens with probability $1/|\mathbb{F}_{p^k}|$.

4.3 Improved C&P IP with LOVE for Arithmetic Circuits

The protocol from Sect. 4.2 communicates 3 field elements per verified multiplication. We now present an alternative multiplication verification procedure, called `AssertMultVec`, that builds on a protocol from Boneh et al. [BBC+19]⁴. `AssertMultVec` simultaneously proves n multiplication instances at the cost of communicating $n + O(\log(n))$ \mathbb{F} -elements. In particular, for \mathbb{F}_p for $p = 2^{61} - 1$ we require around 64.3 bits of communication per multiplication.

Boneh et al. [BBC+19] introduce a logarithmic-sized proof for “parallel-sum” circuits. In a “parallel-sum” circuit, identical subcircuits C' are evaluated in parallel on possibly different inputs, with the sum of the output of each C' being the output of the overall circuit. The high-level idea then is to embed checks for different instances of C' within a single polynomial, allowing the verifier to verify n instances of C' in parallel. This protocol, when letting C' be a multiplication gate, can then be used to simultaneously verify the sum of n multiplications. We call this protocol `AssertDotProduct`.

In more detail, the `AssertDotProduct` protocol works as follows. Suppose P wants to prove that $[z] = \sum_{i \in [n]} [x_i][y_i]$. P begins by defining n polynomials $f_1, \dots, f_{n/2}, g_1, \dots, g_{n/2}$ such that $f_i(j) = x_{(j-1)n/2+i}$ and $g_i(j) = y_{(j-1)n/2+i}$, and then computing $h = \sum_{i \in [n/2]} f_i g_i$. P then commits to h by committing to its coefficients (denoted as $[h]$). V defines its own polynomials f'_i, g'_i over the committed values $[x_{(j-1)n/2+i}]$ and $[y_{(j-1)n/2+i}]$ to check that $\sum_{i \in [n/2]} f'_i g'_i = h$. By Schwartz-Zippel, this can be done by checking that

$$\sum_{i \in [n/2]} f'_i(r)g'_i(r) = h(r) \tag{1}$$

for a random r chosen by V . Here, observe that the evaluation of f'_i, g'_i, h in a public constant r boils down to multiplying the committed coefficients of each polynomial with appropriate powers of r and summing up the result, both of which are local operations. Then, verifying Eq. 1 after fixing r is again a dot product check, although over vectors of length $n/2$, and we can recursively apply `AssertDotProduct` until $n = 1$. Note that only two \mathbb{F}_{p^r} -elements are communicated during one iteration of `AssertDotProduct`: when committing to h and sending r . See Fig. 4 for a formal presentation of the protocol. There, for the base-case of `AssertDotProduct`, we use the multiplication checking procedure from Sect. 4.2.

Given `AssertDotProduct`, we can batch-verify n multiplications as follows:

1. Assume that n tuples $[x_i], [y_i], [z_i]$ have been committed by P .
2. V chooses a randomization factor r that it sends to P .
3. P shows that $\langle r^i[x_i], [y_i] \rangle = \sum_{i \in [n]} r^i[z_i]$. Since r is public, computing $r^i[x_i]$ and $\sum_{i \in [n]} r^i[z_i]$ is local.

This protocol, called `AssertMultVec`, is presented in Fig. 4.

⁴ This approach was recently used in the context of MPC-in-the-head-based ZK protocols [dSGOT21].

$$\text{AssertMultVec}([x_1], \dots, [x_n], [y_1], \dots, [y_2], [z_1], \dots, [z_n]) \Rightarrow \forall i \ x_i y_i = z_i$$

1. $\text{Send}_{[V \rightarrow P]}(r)$ for $r \in_R \mathbb{F}_{p^k} \setminus \{0\}$.
2. $\text{AssertDotProduct}(r^1[x_1], \dots, r^n[x_n], [y_1], \dots, [y_n], \sum_{i \in [n]} r^i[z_i])$.

$$\text{AssertDotProduct}([x_1], \dots, [x_n], [y_1], \dots, [y_n], [z]) \Rightarrow z = \sum_i x_i y_i$$

If $n \leq 2$:

1. For $i \in [n]$: $\text{Mult}([x_i], [y_i]) \rightarrow ([z_i], [a_i], [c_i])$.
2. Run $\text{Send}_{[V \rightarrow P]}(e)$, where $e \in_R \mathbb{F}_{p^k}$.
3. For $i \in [n]$: $\text{AssertMult}([x_i], [y_i], [z_i], [a_i], [c_i], e)$.
4. $\text{AssertZero}(\sum_{i \in [n]} [z_i] - [z])$.

Otherwise:

1. P defines polynomials of least degree $f_1, \dots, f_{n/2}, g_1, \dots, g_{n/2} \in \mathbb{F}_p[X]$ such that for $j \in [2]$: $f_i(j) = x_{(j-1)n/2+i}$, $g_i(j) = y_{(j-1)n/2+i}$.
P defines the polynomial $h = \sum_{i \in [n/2]} f_i g_i \in \mathbb{F}_p[X]$. Note that h has degree ≤ 2 . Let c_0, c_1, c_2 denote the coefficients of h .
2. For $i \in \{0, 1, 2\}$: $\text{Fix}(c_i) \rightarrow [c_i]$.
3. For $i \in [n/2]$: P and V compute (committed) polynomials of least degree $[f'_i]$ and $[g'_i]$ satisfying for $j \in [2]$: $f'_i(j) = [x_{(j-1)n/2+i}]$, $g'_i(j) = [y_{(j-1)n/2+i}]$.
4. Let $[h']$ be the (committed) polynomial defined by the $[c_i]$ values.
5. $\text{Send}_{[V \rightarrow P]}(r)$ where $r \in_R \mathbb{F}_{p^k} \setminus \{0, 1\}$.
6. $\text{AssertZero}(\sum_{i \in [2]} [h'](i) - [z])$.
7. $\text{AssertDotProduct}([f'_1](r), \dots, [f'_{n/2}](r), [g'_1](r), \dots, [g'_{n/2}](r), [h'](r))$.

Fig. 4. Protocols for efficient multiplications. See text for necessary notation.

It is clear that both AssertDotProduct and AssertMultVec are complete and zero-knowledge. The follow theorem, proven in the full version, shows they are also sound.

Theorem 4. *If the protocol AssertMultVec passes, then the input commitments have the required relation except with probability $\frac{n+4 \log n+1}{p^k-2}$*

An alternative version of AssertMultVec with a soundness error that is only logarithmic in n can be achieved as follows:

$\text{AssertMultVec}'([x_1], \dots, [x_n], [y_1], \dots, [y_2], [z_1], \dots, [z_n])$:

1. $\text{Send}_{[V \rightarrow P]}(r_1, \dots, r_n)$ for $r_1, \dots, r_n \in_R \mathbb{F}_{p^k}$.
2. $\text{AssertDotProduct}(r_1[x_1], \dots, r_n[x_n], [y_1], \dots, [y_n], \sum_{i \in [n]} r_i[z_i])$.

One can easily show that $\text{AssertMultVec}'$ has the desired soundness, although at the expense of communicating more random elements from V to P. In practice, one can optimize this by having V choose a random PRG seed that it sends to P, with r_1, \dots, r_n derived deterministically from the seed.

4.4 C&P IP with LOVE for Binary Circuits

The protocol from Sect. 4.3 is agnostic to the underlying field, so we can use $p = 2$ (and large enough k for soundness) to obtain a proof for binary circuits. For $\mathbb{F}_{2^{40}}$ and a batch size of 1 000 000 this requires approximately 1.008 bits per verified AND-gate.

One of the downsides to the batching approach is that it is most efficient for large batches of multiplications. When evaluating a disjunctive branch, however, the size of the batch may be limited by the number of multiplications in the branch. This is because we need to “complete” a batch of multiplications before we can apply the OR-proof. Unfortunately, this smaller batch size increases the per-bit communication cost: as an example, a batch size of 100 requires approximately 10 bits per verified AND-gate.

We now present an alternative approach that can achieve a fixed per-bit communication cost of 9 bits per verified AND-gate. This approach uses *reverse multiplication friendly embeddings* [BMN18, CCXY18] (RFMEs), defined as follows.

Definition 6. A $(k, m)_p$ -RFME is a pair (ϕ, ψ) of linear maps $\phi : \mathbb{F}_p^k \rightarrow \mathbb{F}_{p^m}$ and $\psi : \mathbb{F}_{p^m} \rightarrow \mathbb{F}_p^k$ such that $\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$, where $*$ denotes pairwise multiplication.

Cascudo et al. [CCXY18] showed that for $p = 2$ and $r < 33$, there exist $(3r, 10r - 5)_2$ -RFMEs. Noting that for efficiency we would like as small a field as possible, alongside the requirement of have a statistical security parameter of at least 40, we use $(15, 45)_2$ -RFMEs, and thus work over $\mathbb{F}_{2^{45}}$. Thus, we can verify the multiplication of 15-element binary vectors $[\mathbf{x}]$ and $[\mathbf{y}]$ at the cost of a single multiplication in $\mathbb{F}_{2^{45}}$ as follows. The parties locally compute $[a] \leftarrow \phi([\mathbf{x}])$ and $[b] \leftarrow \phi([\mathbf{y}])$, compute $[c] \leftarrow [a] \cdot [b]$ using the multiplication verification protocol over $\mathbb{F}_{2^{45}}$, and finally locally compute $[z] \leftarrow \psi([c])$. This has a per-multiplication cost of 10 bits per multiplication.

We can do slightly better by having the prover provide the verifier an *advice* vector to help compute $[c]$. Let \mathbf{d} be a binary vector for the linear bijection $f : \mathbb{F}_2^{15} \times \mathbb{F}_2^{30} \rightarrow \mathbb{F}_2^{45}$ such that $f(\mathbf{z}, \mathbf{d}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$. If $[z]$ is provided by the prover, the verifier can *locally* compute $[c]$ by computing $[\mathbf{c}'] \leftarrow f([\mathbf{z}], [\mathbf{d}])$ and then mapping $[\mathbf{c}']$ to its associated element in $\mathbb{F}_{2^{45}}$. The parties can then check that $[c] = [a] \cdot [b]$ as before. Overall this gives a per-bit communication cost of 9 bits. See the full version for more details.

5 Streaming and Non-interactive Proofs via Fiat-Shamir

We now show how to modify our previous constructions for arithmetic circuit satisfiability and disjunctions to support streaming, and also be non-interactive via a variant of the Fiat-Shamir transform [FS87]. We first show how to stream our IPs with LOVE, at the cost of increased round complexity. Then, we show how IPs with LOVE can be transformed into NIZKs (with VOLE preprocessing)

with the Fiat-Shamir transform. To analyze the soundness of this approach, we define a form of *round-by-round soundness* for IPs with LOVE, similar to Canetti et al. [CCH+19], and show that this is satisfied by our constructions.

5.1 Streaming Interactive Proofs

We use the term *streaming* to refer to a protocol where both the prover and verifier algorithms can be run using only a constant amount of memory, independent of the size of the statement and witness. For disjunctive proofs, we relax this to allow $O(m)$ memory, where m is the maximum number of branches in any disjunction. Note that when looking at a C&P IP with LOVE, in addition to requiring a small memory footprint for P and V, we also need that the linear oracle queries can be performed with small memory. It is enough to require that P can compute the result of each oracle query incrementally during the protocol, and with constant memory; when translating the IP with LOVE into a zero-knowledge proof based on VOLE (Sect. 2.2), this ensures that the resulting protocol also has constant memory, since each `AssertZero` can be checked on-the-fly.

Recall that in our protocols for circuit satisfiability, the multiplication gates are all verified in a batch at the end of the computation. This requires storing all commitments created during each multiplication in memory, leading to a memory cost that is linear in the circuit size.

For the more efficient amortized protocol, this drawback seems inherent, however, we can easily avoid it for the simpler protocol from Sect. 4.2, by checking multiplications on-the-fly using an independent random challenge from V for each multiplication. The change is very simple, and for completeness, shown in the modified multiplication sub-protocol below.

Streaming Mult($[x_i], [y_i]$): To evaluate the i -th multiplication gate:

1. Run `Random`(\mathbb{F}_p) to get $[r_i]$ and `Random`(\mathbb{F}_{p^k}) to get $[r'_i], [a_i]$.
2. Run `Fix`($x_i y_i \rightarrow [z_i]$), and `Fix`($a_i y_i \rightarrow [c_i]$).
3. Run `Send`_[V→P]($e_i \leftarrow \mathbb{F}_{p^k}$).
4. Run `Reveal`($[\varepsilon]$), where $[\varepsilon] = e_i [x_i] - [a_i]$.
5. Run `AssertZero`($e_i [z_i] - [c_i] - \varepsilon_i [y_i]$).

For the soundness of this protocol, following the exact same analysis as in Sect. 4.2, we get a soundness error of p^{-k} , due to the random choice of each challenge e_i . In Sect. 5.3, we show that this protocol also satisfies *round-by-round soundness*, implying that it can be made non-interactive using Fiat-Shamir.

5.2 Batching AssertZero with Constant Memory

Recall from Sect. 2 that often, it is useful to combine all the `AssertZero` statements (that is, linear oracle queries) of an IP with LOVE into just one check, by batching them together at the end. However, just as with our original circuit evaluation protocol, this is not amenable to constant memory for streaming

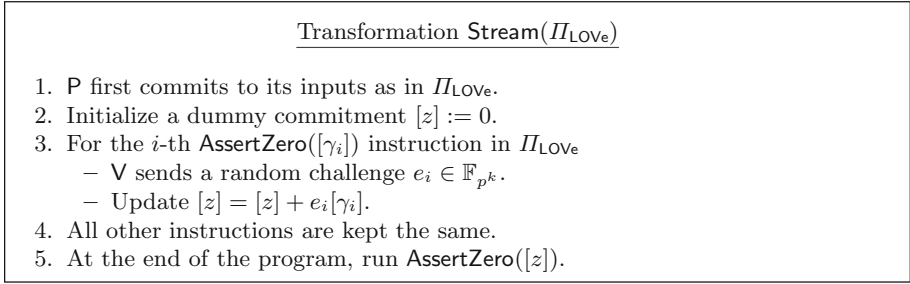


Fig. 5. The transformation to batch AssertZero in a streamable manner.

algorithms. Instead, in Fig. 5 we give an alternative transformation, which transforms any C&P IP with LOVE Π_{LOVe} to have just one AssertZero , *without* storing all intermediate values.

The idea is that, instead of taking a combination of all $\text{AssertZero}'s$ at the end, we can compute this combination in an incremental manner. At each AssertZero on input $[\gamma]$, we take a random challenge e and add $e \cdot [\gamma]$ to a running state $[z]$. At the end of the computation, to verify that all the γ values were zero, we simply run AssertZero on $[z]$. Since the challenge e is only sampled *after* the value being checked for zero was committed, it holds from the argument for the batching method from Sect. 2 that cheating in this check requires guessing a random challenge, so this transformation only increases the soundness error by p^{-k} .

5.3 Round-by-Round Soundness for IPs with LOVE

The intuition behind the definition of *Round-by-round soundness* is that in any given round of a protocol Π , one can define a function State , which, given the current transcript of Π , outputs a bit indicating whether Π 's execution will fail. We require that, if State predicts failure in some round i , then State also predicts failure in round $i + 1$, with high probability over the verifier's random challenge.

Previously, round-by-round soundness has been defined for standard interactive proofs [CCH+19], without any form of oracle queries. Below is our modified definition, tailored to IPs with LOVE. Note that unlike the Zero Knowledge setting, where State 's inputs are publicly known, here we give the State function also the oracle string π as input; without this, there would be no easy way for the State algorithm to simulate whether a given oracle query to π would succeed or not. Note also that since we assume Π is public-coin, the oracle query inputs (z, y) are all computable given the complete transcript, so they are also known to State .

Definition 7. *Let Π be a t -round, public-coin IP-LOVe for a relation \mathcal{R} . We say that Π has round-by-round soundness error ϵ if there exists a deterministic (not necessarily efficient) function State that takes input an instance x , proof π and partial transcript \mathcal{T} , and outputs accept or reject , such that the following properties hold:*

1. If $x \notin L$, then $\text{State}(x, \pi, \emptyset) = \text{reject}$.
2. If $\text{State}(x, \pi, \mathcal{T}) = \text{reject}$ for a partial transcript \mathcal{T} up to round $h \in [t]$, then for every potential prover message \mathbf{a}_h ,

$$\Pr_{e_{h+1} \leftarrow \mathbb{F}^{\vee_{h+1}}} [\text{State}(x, \pi, \mathcal{T} \parallel \mathbf{a}_h \parallel \mathbf{e}_{h+1}) = \text{accept}] \leq \epsilon$$

3. For any halting transcript \mathcal{T} , if $\text{State}(x, \pi, \mathcal{T}) = \text{reject}$ then \mathcal{V} rejects.

Round-by-round soundness of our protocols for circuit satisfiability.

In the full version, we show that our IPs with LOVE for circuit satisfiability, including the streamable protocol from Sect. 5.1–5.2, and the efficient batched multiplication protocol from Sect. 4.3 satisfy round-by-round soundness. We also show that the same holds for our stacking protocol from Sect. 3.

Roughly speaking, for our circuit satisfiability protocol, the `State` algorithm takes as input the proof string π , so can immediately extract the witness and try to verify whether the statement is true. In later rounds, `State` also checks whether the prover’s messages are inconsistent with π and the verifier’s challenges, and changes to `reject` if so. A similar strategy works in all our protocols to show that round-by-round soundness holds.

Soundness of Fiat-Shamir for IPs with LOVE. We now show that the Fiat-Shamir transformation, when applied to a zero-knowledge proof built from VOLE and an IP with LOVE, is sound if the underlying IP with LOVE satisfies round-by-round soundness.

For this, we follow the VOLE-based protocol from Sect. 2.2, while replacing the verifier’s random challenges with outputs of a random oracle. We use a slightly augmented VOLE functionality, denoted $\mathcal{F}_{\text{VOLE}+\text{id}}$, which additionally samples a random identifier $\text{id} \in \{0, 1\}^\lambda$, and gives this to both parties after receiving their input. We feed this into the random oracle, which binds the statement and proof to this instance. The result we obtain is similar to the FS transform for interactive oracle proofs [BCS16], with the differences that (1) we start from IPs with LOVE using VOLE preprocessing, and (2) we assume round-by-round soundness, which is a stronger property than state-restoration soundness from [BCS16], but we find it simpler to work with. We prove the following theorem in the full version.

Theorem 5. *Let Π_{LOVe} be a t -round, 1-query, public-coin IP-LOVe for relation \mathcal{R} with round-by-round soundness ϵ , which is also complete and zero-knowledge. Then, the compiled protocol $\Pi_{\text{NIZK}}^{\text{VOLE}}$ in Fig. 6 is a non-interactive zero-knowledge proof in the $\mathcal{F}_{\text{VOLE}+\text{id}}$ -hybrid model, with soundness error at most*

$$p^{-k} + \epsilon t + Q(\epsilon + 2/|\mathcal{C}| + 2^{-\lambda})$$

where Q is the number of random oracle queries made by a malicious prover, and $|\mathcal{C}|$ is the size of the smallest challenge set in any given round of Π .

Furthermore, if Π_{LOVe} is a proof of knowledge, then so is $\Pi_{\text{NIZK}}^{\text{VOLE}}$.

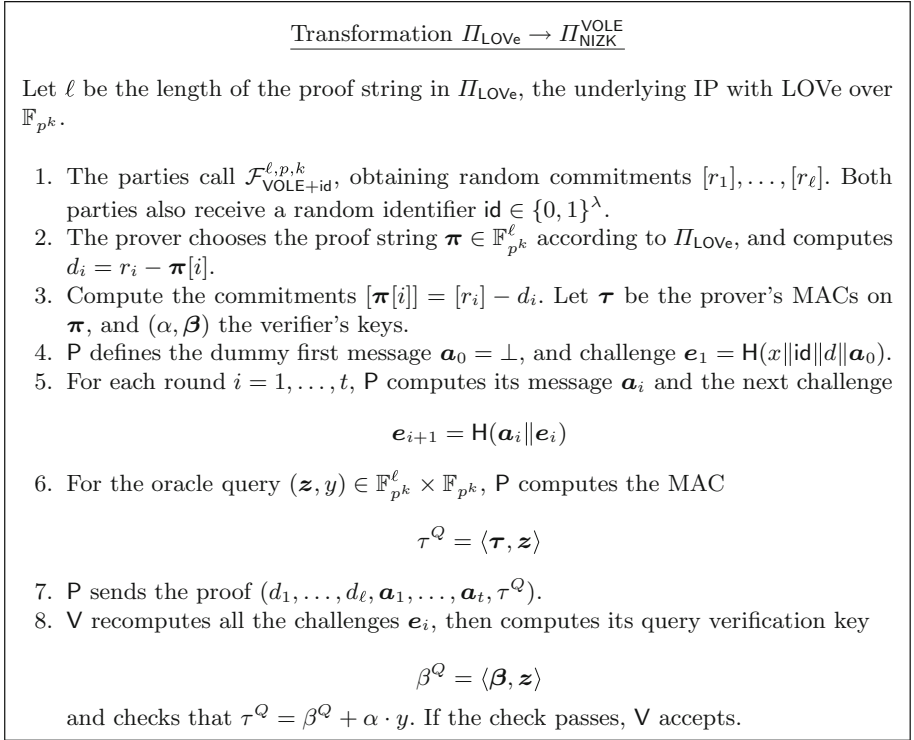


Fig. 6. NIZK from VOLE and IP with LOVe.

6 Implementation and Evaluation

We have implemented the online protocol of Mac'n'Cheese with the batched multiplication approach of Sect. 4.3 in the Rust programming language. Our implementation achieves a computational security of 128 bits and a statistical security of ≥ 40 bits. Our implementation supports pluggable VOLE backends. The backend that we use, at present, is a “dummy” backend which (insecurely) generates random MACs by using a pre-shared seed with a PRNG.

Streaming. To facilitate streaming, our implementation does not view its input as an explicit circuit graph. Instead, the proof statement is lazily built-up by a series of function invocations. As a result, we get reduced memory consumption (for free) as temporary values get automatically freed when they are no longer in-scope. In order to stream a two-way disjunction, both branches of the disjunction must be interleaved (otherwise the prover would be forced to either buffer the entirety of a branch, or reveal which branch is ‘true’). To achieve this interleaving, we leverage stackful coroutines to (cheaply) concurrently execute both branches.

Concurrency. Despite running a multi-round interactive protocol (without extensive use of the Fiat-Shamir transform), we are still able to achieve high-performance even over a high-latency, throughput-limited network link. We reach

this result by running one thread which exclusively sends data from the prover to the verifier to Fix MACs to prover-private values. Once this thread has Fixed a batch of MACs, it submits them to one of many background threads to verify the assertions on these MACs. Each background thread has its own unique connection between the prover and the verifier, so can independently wait for a response from the other party. As a result, we can provision a large number of background threads (which will spend most of their time waiting for the network, rather than running computation) to mitigate the latency effects of running our multi-round protocol over a network. In addition, this design allows us to leverage multiple cores independent of the circuit structure.

6.1 Evaluation

We benchmarked our implementation for \mathbb{F}_{240} (for boolean circuits) and \mathbb{F}_{261-1} (for arithmetic circuits). Unless otherwise specified, all benchmarks were run between two machines: a laptop (2018 MacBook Pro with 8 logical cores and 16 GB of RAM) on the east coast of the U.S., and a server (40 Intel Xeon Silver 4114 cores operating at 2.20 GHz) on the west coast of the U.S. The network had an average latency of 95 ms and an average bandwidth of 31.5 Mbps. All numbers are the average of at least four runs of the given experiment.

As noted above, these results *do not* include the cost of VOLE. We are in the process of integrating the VOLE protocol of Weng et al. [WYKW20], but do not believe this will have a large impact on the overall running time and communication cost given that a single VOLE for \mathbb{F}_{261-1} can be generated in 85 ns at a communication cost of 0.42 bits [WYKW20, Table 4], and can be largely precomputed.

As mentioned above, our implementation is multi-threaded, and in order to reduce communication latency we pipeline processing as much as possible. We use 50 threads for all of our experiments, although we note that the CPU utilization on both the prover and verifier never exceeds 226% (where the maximum possible utilization is the number of cores times 100%). We reiterate that our verifier was run on a commodity laptop, and while we use a large number of threads, this does *not* equate to extremely high CPU utilization.

Microbenchmarks. Using a multiplication batch size of 1 000 000, Mac’n’Cheese achieves a per multiplication cost of approximately 144 ns for \mathbb{F}_{240} and 1.5 μ s for \mathbb{F}_{261-1} . This equates to 6.9 million multiplications per second (mmps) for \mathbb{F}_{240} , and 0.6 mmps for \mathbb{F}_{261-1} . We found that the main limiter in the arithmetic case was bandwidth, and thus also ran our microbenchmarks locally (run on the Location B server), achieving a per multiplication cost of 141 ns (7.0 mmps) for \mathbb{F}_{240} and 276 ns (3.6 mmps) for \mathbb{F}_{261-1} .

Comparison to QuickSilver. We briefly compare to QuickSilver [YSWW21]. Recall that QuickSilver requires only a single field element per multiplication and requires only 3 rounds (cf. Table 1), but does not support communication-optimized disjunctions. When run on localhost within an Amazon EC2 instance,

Table 2. Performance results for disjunctions. The “Branches” column denotes the number of branches, where each branch contains 1 billion AND gates. The “Local” column denotes the time to locally compute the circuit in-the-clear, and provides a rough lower bound of performance. The “Verify” column denotes the time to verify the ZK proof. The “Comm. Increase” column denotes the amount of communication increase from the baseline of 124 MB required in the single-branch case.

Branches	Local (seconds)	Verify (seconds)	Comm. Increase (bytes)
1	34	139	—
2	81	307	+25
4	163	568	+50
8	327	1254	+75

QuickSilver achieves 7.6 mmps for boolean and 4.8 mmps for arithmetic when utilizing 1 thread, and 15.8 mmps for boolean and 8.9 mmps for arithmetic when utilizing 4 threads [YSWW21, Table 2]. While it is hard to make an apples-to-apples comparison here, this does suggest that QuickSilver is slightly faster, albeit at the expense of communication-optimized disjunctions. Thus, the choice of QuickSilver versus Mac’n’Cheese may come down to the characteristics of the input circuit.

Disjunctions. We also explored the effect our disjunction optimization has on the communication cost. We did so by comparing a proof of a boolean circuit containing 1 billion multiplication gates to using a boolean circuit containing two or more branches each containing 1 billion gates⁵. See Table 2 for the results.

The overall communication in all cases was essentially 124 MB: the OR proof added only an additional $25 \log(m)$ bytes, where m denotes the number of branches. In terms of overall running time, we see an increase with the overall *size* of the circuit. This is due to the fact that the prover still needs to do the entire computation, and for this particular example bandwidth is *not* the bottleneck. The table also reports the time required to simply *evaluate* the circuit locally—this presents a reasonable lower bound for Mac’n’Cheese. We find that in all cases, Mac’n’Cheese takes less than $4.08\times$ the cost of locally evaluating the circuit.

Acknowledgments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the

⁵ In more detail, the branched circuit contained one branch computing 150 000 iterations of AES (960 million multiplication gates) and the other branch computing 45 000 iterations of SHA-2 (1.002 billion multiplication gates). The non-branched circuit only ran the SHA-2 portion of the aforementioned circuit.

Defense Advanced Research Projects Agency (DARPA). Distribution Statement “A” (Approved for Public Release, Distribution Unlimited).

References

- [AHIV17] Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Liger: lightweight sublinear arguments without a trusted setup. In: ACM CCS 2017. ACM Press, October/November 2017
- [BBC+19] Boneh, D., Boyle, E., Corrigan-Gibbs, H., Gilboa, N., Ishai, Y.: Zero-knowledge proofs on secret-shared data via fully linear PCPs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 67–97. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_3
- [BBHR19] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 701–732. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_23
- [BCG+13] 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, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_6
- [BCG+14] Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, May 2014
- [BCG+19a] Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: ACM CCS 2019, November 2019
- [BCG+19b] 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 (2019). https://doi.org/10.1007/978-3-030-26954-8_16
- [BCGI18] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: ACM CCS 2018. ACM Press, October 2018
- [BCS16] Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) TCC 2016, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2
- [Bea92] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
- [BG90] Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_19
- [BMN18] Block, A.R., Maji, H.K., Nguyen, H.H.: Secure computation with constant communication overhead using multiplication embeddings. In: Chakraborty, D., Iwata, T. (eds.) INDOCRYPT 2018. LNCS, vol. 11356, pp. 375–398. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05378-9_20

- [CCH+19] Canetti, R., et al.: Fiat-Shamir: from practice to theory. In: 51st ACM STOC. ACM Press, June 2019
- [CCXY18] Cascudo, I., Cramer, R., Xing, C., Yuan, C.: Amortized complexity of information-theoretically secure MPC revisited. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 395–426. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_14
- [CD97] Cramer, R., Damgård, I.: Linear zero-knowledge - a note on efficient zero-knowledge proofs and arguments. In: 29th ACM STOC, May 1997
- [CD98] Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055745>
- [CDS94] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
- [DIO21] Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-point zero knowledge and its applications. In: Information-Theoretic Cryptography (ITC) 2021 (2021)
- [dSGOT21] de Saint Guilhem, C.D., Orsini, E., Tanguy, T.: Limbo: efficient zero-knowledge MPCitH-based arguments. Cryptology ePrint Archive, Report 2021/215 (2021)
- [FNO15] Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 191–219. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_7
- [FS87] Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
- [GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: 19th ACM STOC. ACM Press, May 1987
- [HK20a] Heath, D., Kolesnikov, V.: Stacked garbling. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 763–792. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_27
- [HK20b] 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 (2020). https://doi.org/10.1007/978-3-030-45727-3_19
- [JKO13] Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: ACM CCS 2013. ACM Press, November 2013
- [Kol18] Kolesnikov, V.: Free IF: how to omit inactive branches and implement \mathcal{S} -universal garbled circuit (almost) for free. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 34–58. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_2
- [KOS16] Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: ACM CCS 2016. ACM Press, October 2016

- [WYKW20] Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. Cryptology ePrint Archive, Report 2020/925 (2020). <https://eprint.iacr.org/2020/925>
- [YSWW21] Yang, K., Sarkar, P., Weng, C., Wang, X.: Quicksilver: efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. Cryptology ePrint Archive, Report 2021/076 (2021)
- [ZRE15] Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_8



Time- and Space-Efficient Arguments from Groups of Unknown Order

Alexander R. Block¹(✉), Justin Holmgren², Alon Rosen³, Ron D. Rothblum⁴,
and Pratik Soni⁵

¹ Purdue University, West Lafayette, USA
block9@purdue.edu

² NTT Research, Sunnyvale, CA, USA
justin.holmgren@ntt-research.com

³ IDC Herzliya, Herzliya, Israel
alon.rosen@idc.ac.il

⁴ Technion, Haifa, Israel

rothblum@cs.technion.ac.il

⁵ Carnegie Mellon University, Pittsburgh, USA
psoni@andrew.cmu.edu

Abstract. We construct public-coin time- and space-efficient zero-knowledge arguments for **NP**. For every time T and space S non-deterministic RAM computation, the prover runs in time $T \cdot \text{polylog}(T)$ and space $S \cdot \text{polylog}(T)$, and the verifier runs in time $n \cdot \text{polylog}(T)$, where n is the input length. Our protocol relies on hidden order groups, which can be instantiated with a trusted setup from the hardness of factoring (products of safe primes), or without a trusted setup using class groups. The argument-system can heuristically be made non-interactive using the Fiat-Shamir transform.

Our proof builds on DARK (Bünz et al., Eurocrypt 2020), a recent succinct and efficiently verifiable *polynomial commitment scheme*. We show how to implement a variant of DARK in a time- and space-efficient way. Along the way we:

1. Identify a significant gap in the proof of security of DARK.
2. Give a non-trivial modification of the DARK scheme that overcomes the aforementioned gap. The modified version also relies on significantly weaker cryptographic assumptions than those in the original DARK scheme. Our proof utilizes ideas from the theory of integer lattices in a novel way.
3. Generalize Pietrzak's (ITCS 2019) proof of exponentiation (PoE) protocol to work with general groups of unknown order (without relying on any cryptographic assumption).

In proving these results, we develop general-purpose techniques for working with (hidden order) groups, which may be of independent interest.

1 Introduction

Significant overhead in prover efficiency is the main roadblock standing between zero-knowledge proofs and widespread deployment. While there has been exten-

sive work on optimizing the *running time* of the prover, much less attention has been drawn to the *space (or memory) usage*. In particular, most protocols in the literature suffer from the drawback that memory consumption by the prover is exceedingly large: computations that take time T and space S to compute directly, require the prover to invest $\Omega(T)$ space in order to prove correctness (with some notable exceptions [9–11, 13, 31, 44] to be discussed shortly). Moreover, due to the way that modern memory architectures work, large memory usage also inevitably leads to more cache misses and slower runtime. Thus, space efficiency of the prover is a severe bottleneck to enabling zero-knowledge proofs for large-scale complex computations.

The recent work of Block et al. [11] constructed the first *publicly verifiable*¹ zero-knowledge proofs (under standard cryptographic assumptions) in which the prover is efficient both in terms of time and space. In more detail, for every **NP** relation R , for which membership can be computed in time T and space S , the prover (given as input the instance and corresponding witness) can be implemented in time $T \cdot \text{poly}(\lambda, \log T)$ and space $S \cdot \text{poly}(\lambda, \log T)$ and the verifier can be implemented in time roughly $T \cdot \text{poly}(\lambda, \log T)$, where here and throughout this work λ denotes the security parameter. The fact that verification takes $\Omega(T)$ time is a significant drawback of this protocol and precludes applications like delegation of computation.

1.1 Our Results

In this work we overcome the main disadvantage of the work of Block et al. by constructing zero-knowledge proofs with a time- and space-efficient prover *and* poly-logarithmic verification. For this result we rely on groups of unknown order, which are discussed immediately after the statement of Theorem 1.1.

Theorem 1.1 (Informally Stated, see Theorem 4.1). *Assume that there exists a group for which the hidden order assumption holds. Then, every **NP** relation that can be verified by a time T and space S RAM machine has a public-coin zero-knowledge argument-system in which the prover, given as input the instance x and witness w , runs in time $T \cdot \text{poly}(\lambda, \log T)$ and uses space $S \cdot \text{poly}(\lambda, \log T)$. The verifier runs in time $|x| \cdot \text{poly}(\lambda, \log T)$, the communication complexity is $\text{poly}(\lambda, \log T)$ and the number of rounds is $O(\log T)$.*

The argument-system uses a common reference string, which is simply a description of the hidden-order group \mathbb{G} and a random element $g \in \mathbb{G}$.

As usual, the protocol can heuristically be made *non-interactive* by applying the Fiat-Shamir [29] transform. It is also worth noting that a result similar to Theorem 1.1 was not known even *without* the zero-knowledge requirement.

¹ Public verifiability has emerged as a central requirement for proof-systems. In a nutshell it means that anyone who possesses the proof-string can verify its correctness (while possibly also requiring access to a common reference string). We mention that time- and space-efficient protocols that are either *privately-verifiable* or based on non-standard computational assumptions were previously known. See Sect. 1.2.

As for the assumption that we use, the *hidden order assumption* for a group \mathbb{G} states that given a random group element $g \in \mathbb{G}$ it is computationally infeasible to find (any multiple of) the order of g . For example, assuming the hardness of factoring N which is a product of two safe primes, the group \mathbb{Z}_N^* , is a hidden order group. Therefore, our scheme can be instantiated assuming the hardness of factoring (products of safe primes).

Lately there has been much interest in *public-coin* hidden order groups which means that the description of the group can be generated without a trusted party (aka a *transparent* setup). This is not known for the factoring based group (since one needs to be able to generate a hard instance for factoring without using private coins). However, as pointed out in [21, 26], class groups of an imaginary quadratic field are a candidate public-coin hidden order group. Since our common reference string only includes a description of the group and a random element, using class groups we obtain a protocol that does not require a trusted setup.

Time- and Space-efficient Polynomial Commitments. Theorem 1.1 is derived from a new polynomial-commitment scheme that we construct, based on a prior scheme due to Bünz et al. [21]. Roughly speaking, a *polynomial-commitment scheme* allows Alice to commit to a low degree polynomial P so that later Bob can ask her for evaluations $P(x)$ along with proofs that the supplied values are indeed consistent with her commitment (see Sect. 3.4 for the formal definition). Polynomial commitments have drawn significant attention recently (see Sect. 1.2), especially due to their use in compiling ideal model information-theoretic proof-systems into real-world protocols. Most works use polynomial commitments in order to obtain shorter proof sizes. In contrast, following [11], we use polynomial commitments to enable a small space (and time) implementation of the prover. We believe that this aspect of polynomial commitments will be a key enabler of large-scale zero-knowledge proofs.

For simplicity, and since it suffices for proving Theorem 1.1, we focus on polynomial commitments for *multilinear*² polynomials $P : \mathbb{F}^n \rightarrow \mathbb{F}$, where \mathbb{F} is a prime order field. Following [11], we consider polynomial commitments in a streaming model, in which the committer are given (*multi-pass*) *streaming access* to the representation of the polynomial; in our case, the restriction of the multilinear polynomial to the Boolean hypercube. This streaming model is motivated by the fact that when using the commitment scheme to construct an efficient argument-system, the prover commits to a transcript of the computation - which can indeed be generated in a streaming manner in small space.

In order to construct their time- and space-efficient arguments, [11] first construct a polynomial commitment scheme for multilinear polynomials in which the prover runs in quasi-linear time (in the description of the polynomial, which is of size $2^n \cdot \log(|\mathbb{F}|)$) and logarithmic space. However, the *verifier* for their evaluation proof also runs in time that is linear in the size of the polynomial. This is the core reason that the argument-system constructed in [11] does not achieve *sub-linear* verification. In contrast, we give a polynomial commitment scheme in which the prover is time- and space-efficient and verification *is* poly-logarithmic.

² Recall that a multi-variate polynomial is multilinear if its degree in each variable is at most 1.

Theorem 1.2 (Informally Stated, see Theorem 4.2). *Assume that there exists a group for which the hidden order assumption holds. Then, there exists a polynomial commitment scheme for multilinear polynomials $P : \mathbb{F}^n \rightarrow \mathbb{F}$ over a prime order field \mathbb{F} (of size $|\mathbb{F}| \leq 2^{\text{poly}(n)}$) with the following efficiency properties:*

1. *Commitment and evaluation proofs can be computed in time $2^n \cdot \text{poly}(n, \lambda)$ and space $n \cdot \text{poly}(\lambda)$, given multi-pass streaming access to the evaluations of P on the Boolean hypercube.*
2. *The communication complexity and verification time are both $\text{poly}(n, \lambda)$.*

Similarly to Theorem 1.1, the commitment scheme is defined relative to a reference string containing the description of the hidden order group and a random group element.

Theorem 1.1 follows from Theorem 1.2 using techniques from the work of Block et al. [11]. Namely, we use a time- and space-efficient *polynomial interactive oracle proof*³ (polynomial IOP), constructed in [11] (based on the 2-prover MIP of [13]). We then compile this polynomial IOP into an argument-system using the polynomial commitment of Theorem 1.2 in the natural way: namely, rather than sending polynomials in the clear, the prover simply commits to them and later proves correctness of evaluations queries. This compilation results in a *succinct* argument, which can be made zero-knowledge (while preserving time- and space-efficiency) using standard techniques [4] (see [12] for details).

Our proof of Theorem 1.2 builds on a recent remarkable polynomial-commitment scheme called DARK (for Diophantine Argument of Knowledge), due to Bünz et al. [21]. This polynomial commitment scheme was the first such scheme to achieve logarithmic size proofs and verification time.

We make several significant improvements to the DARK scheme.

1. **Identifying and Bypassing a Gap in DARK:** We identify a gap in the security proof of [21]. We elaborate on this gap in Sect. 2.2. We emphasize that we do not know whether this gap can lead to an attack on the DARK scheme. Nevertheless, we find this gap to be significant and in particular we do not know how to fix their security proof. We mention that we have been informed [23] that the same gap was discovered independently by the authors of [21].

To obtain our polynomial commitment scheme, we therefore make a non-trivial modification of the DARK scheme and show that this modification suffices to prove security. Our security proof relies on a new lemma on the existence of integral inverses for uniformly random rectangular binary matrices, which we prove. Our proof is based on ideas from the mathematical theory

³ A polynomial IOP is defined similarly to a (public-coin) interactive proof, except that in every round the prover is allowed to send the truth table of a large polynomial, and the verifier can query a few points from each polynomial. The notion was proposed concurrently in [21] and [24]. Essentially the same notion appears also in [38] (called *Probabilistically Checkable Interactive Proof w.r.t. Encoded Provers* therein).

of integer lattices which, to the best of our knowledge, have not been used before in this context.⁴ See Sect. 2.3 for details.

2. **Improved Assumptions and Simplicity:** Setting aside the gap in the security proof, we also significantly improve the assumptions that the DARK scheme relies on. The improvement in assumptions stems from a *simpler* (and conceptually more appealing) extraction procedure that we describe. This improvement applies to the two main variants of the DARK scheme. In more detail:
 - (a) The first variant of the original DARK scheme uses RSA groups, while relying on the *strong RSA assumption* and the *adaptive root assumption*. The former assumption, while not new, is relatively strong, whereas the latter is a new assumption, due to Wesolowski [46], which is not yet well understood (note that both assumptions are known to hold in the generic group model [16, 27]).
In contrast, when instantiating our scheme in this setting, we only need to rely on the hardness of factoring (products of safe primes).
 - (b) In order to obtain an *unstructured* common random string, Bünz et al. also give a construction that uses class groups of an imaginary quadratic field. This construction relies on both the aforementioned adaptive root assumption (for class groups) *and* a new assumption that they introduce on class groups called the 2-strong RSA assumption. The class-group based construction is also more complex than their construction using RSA groups.
In contrast, our construction works equally well for both groups and we can instantiate it using class groups while assuming only the hidden order assumption (which is weaker than the adaptive root assumption [15]). See also [1, 43] for a comparison between these assumptions.
3. **Small Space Polynomial Commitments:** We show that the commitment and evaluation protocols in (our variant of) the DARK scheme can be implemented in time roughly $\tilde{O}(2^n)$ (i.e., quasi-linear in the description of the polynomial) *and* space $\text{poly}(n)$ (i.e., poly-logarithmic in the description), given (multi-pass) streaming access to the evaluations of the polynomial on the Boolean hypercube. Crucially, (and in contrast to the scheme of [11]) the verifier in our evaluation proofs runs in time $\text{poly}(n)$. See Sect. 2.4 for the ideas underlying our space-efficient implementation.
4. **Statistical Proof of Exponentiation over General Groups:** We improve and generalize a recent elegant *proof-of-exponentiation* protocol due to Pietrzak [37]. In a **proof of exponentiation** protocol, the goal is for the prover to convince the verifier that the triplet $(g, h, T) \in \mathbb{G} \times \mathbb{G} \times \mathbb{N}$ satisfies the relation $h = g^{2^T}$, where \mathbb{G} is a group of unknown order.⁵ Pietrzak constructs such a protocol in which the prover runs in time roughly T and the verifier

⁴ We emphasize that we use lattice theory to show that our *group* based construction is secure. In particular all of our hardness assumptions are group based.

⁵ Since the order of \mathbb{G} is not known, one cannot simply compute 2^T modulo the group order and then exponentiate.

runs in time roughly $\log(T)$ (which is exponentially faster than the direct computation via repeated squaring). Pietrzak uses his protocol to construct a simple *verifiable delay function* [14], based on the Time-Lock puzzles of Rivest, Shamir and Wagner [39].

Pietrzak’s protocol is designed specifically for the group QR_N^+ of (signed) quadratic residues modulo an integer N , which is the product of two safe primes. Pietrzak [37, Section 6.1] points out that the protocol can also be extended to class groups, but with two caveats. First, this extension is only *computationally* sound and second, it requires an additional assumption from the class group (namely, that it is hard to find elements of small order). This is in contrast to Pietrzak’s protocol for QR_N^+ which provides statistical security and without relying on any assumption. We note that a different protocol, due to Wesolowski [46], gives a proof of exponentiation over groups in which the adaptive root assumption holds (which plausibly includes class groups), but also only achieves computational soundness and requires a (strong) hardness assumption. Wesolowski’s protocol is used as sub-routine within the DARK scheme.

As an additional contribution, which we find to be of independent interest, we show a modification of Pietrzak’s protocol that works over *general* groups of unknown order (including class groups) while preserving *statistical* security and without relying on any assumption. By replacing Wesolowski’s protocol within the DARK scheme with our new extension, we obtain that the evaluation proofs for our polynomial commitment are *proofs* (rather than arguments) of knowledge.

1.2 Additional Related Works

Polynomial Commitments. Polynomial commitment schemes were introduced by Kate et al. [32]. As discussed above, such commitments allow one to commit to a polynomial and later answer evaluation queries while proving consistency with the commitment.

There are several variants of polynomial commitments include privately verifiable schemes [32, 36], publicly-verifiable schemes with trusted setup [21], and zero-knowledge schemes [47]. More recently, much focus has been on obtaining publicly-verifiable schemes without a trusted setup [5, 7, 17, 21, 33, 34, 42, 45, 47, 48]. In all but one prior work, the space complexity of the committer is proportional to the description size of the polynomial. The only exception is the aforementioned work of Block et al. [11] who build a commitment scheme for multilinear polynomials (based on [17, 19]), where the committer’s space complexity is *poly-logarithmic* in the description of the polynomial, assuming that the committer is given multi-pass streaming access to its description. As mentioned above, a key drawback of their work is that the verification is linear in the size of the polynomial.

Lastly, we mention that classical works on low degree testing (à la [40]) as well as more recent works [5, 6, 8] can be used to construct polynomial-commitments

by Merkle hashing the entire truth table of the polynomial (and using a self-correction procedure or protocol).

Privately Verifiable Proofs. The question of constructing proof systems in which the prover is efficient both in terms of time *and* space was first raised by Bitansky and Chiesa [10], who constructed a time- and space-efficient (or in their terminology *complexity preserving*) interactive argument for any problem in **NP** based on *fully homomorphic encryption*. Holmgren and Rothblum [31] constructed *non-interactive* time- and space-efficient arguments for **P** based on the (sub-exponential) learning with errors assumption. The protocols of [10,31] are *privately verifiable*, meaning that only a designated verifier (who knows the randomness used to sampled the verifier messages) is able to verify the proof.

Proofs by Recursive Composition. An alternative approach to *publicly verifiable* time- and space-efficient arguments is by recursively composing SNARKs for **NP** [9,44]. Recursive composition requires both the prover and verifier to make non-black-box usage of an “inner” verifier for a different SNARK, leading to large computational overhead. Several recent works [18,20,25] attempt to solve the inefficiency problems with recursive composition, but at additional expense to the underlying cryptographic assumptions. In particular, these works rely on hash functions that are modeled as random oracles in the security proof, despite being used in a *non-black-box* way by the honest parties. Security thus cannot be reduced to a simple computational hardness assumption, even in the random oracle model. Moreover, the practicality of the schemes crucially requires usage of a novel hash function (e.g., Rescue [2]) with algebraic structure designed to maximize the *efficiency* of non-black-box operations. Such hash functions have endured far less scrutiny than standard SHA hash functions, and the algebraic structure could potentially lead to a security vulnerability.

We also mention a recent work of Ephraim et al. [28] which uses recursive composition to address the related question of implementing the prover in small *depth* (i.e., parallel time).

Multi-Prover Proofs. Block et al. [11] gave the first *publicly-verifiable* time- and space-efficient arguments for **NP** but as noted above (and in contrast to Theorem 1.1), the verification time is linear in the computation. Bitansky and Chiesa [10], as well as Blumberg et al. [13], construct time- and space-efficient *multi-prover* interactive proof, that is, soundness only holds under the assumption that the provers do not collude. Justifying this assumption in practice seems difficult and indeed multi-prover interactive proofs are usually only used as building blocks toward more complex systems.

1.3 Organization

We give overviews of our proof techniques in Sect. 2. Preliminaries are in Sect. 3. In Sect. 4 we formally state our results and in Sect. 5 we describe our polynomial commitment scheme. The rest of the technical sections are deferred to the full version [12].

2 Technical Overview

We start, in Sect. 2.1 with an exposition of (a variant of) the DARK polynomial commitment scheme of [21]. Then, in Sect. 2.2 we describe a gap in their security proof. In Sect. 2.3 we show how to modify their protocol in order to resolve this gap (and simultaneously simplify the extraction procedure and relax the cryptographic assumptions). Then, in Sect. 2.4 we describe our small space implementation and lastly, in Sect. 2.5 we describe our improved proof of exponentiation protocol.

2.1 Overview of the DARK Scheme

We start with an overview of the DARK polynomial commitment scheme. The main scheme constructed in [21] was for *univariate* polynomials. However, for our applications it will be more useful to consider a variant of their scheme for (multi-variate) *multilinear* polynomials.⁶ We emphasize that the gap in the security proof (to be discussed shortly) also applies to the original DARK scheme.

DARK Commitments: Encoding Polynomials by Large Integers. Let $\mathbb{F} = \mathbb{F}_p$ be a finite field of prime order p . Recall that a multilinear polynomial $P : \mathbb{F}^n \rightarrow \mathbb{F}$ can be specified by its evaluations on the Boolean hypercube. Thus, in order to commit to the polynomial P , we will look at the sequence of values $(P(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n}$. In order to commit to this sequence Bünz et al. construct a large integer $\mathcal{Z}(P)$ that encodes it, by looking at this sequence as a base q representation of an integer, for some $q \gg p$. That is, $\mathcal{Z}(P) = \sum_{\mathbf{b} \in \{0,1\}^n} q^{\mathbf{b}} \cdot P(\mathbf{b})$, where \mathbf{b} is interpreted as an integer in the natural way.

The commitment to the polynomial P is simply $c = g^{\mathcal{Z}(P)}$, where g is a random element of the hidden-order group \mathbb{G} specified as part of the CRS. We say that the integer Z is *consistent* with the multilinear polynomial P if, looking at the base q representation of Z , and *reducing each digit modulo p* , we get the sequence $(P(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n}$. Observe that since $q \gg p$, there are many integers Z that are consistent with a given polynomial P (where one of these integers is $\mathcal{Z}(P)$). Nevertheless, the commitment is *binding* since finding two different integers that are consistent with the same commitment reveals a multiple of the order of g , which we assumed is computationally infeasible.

We will rely on the fact that this commitment scheme is homomorphic, in the following sense: given integers Z_1 and Z_2 that are consistent with the polynomial P_1 and P_2 , and have sufficiently small digits in their base q representation, it holds that $Z_1 + Z_2$ is consistent with the polynomial $P_1 + P_2 \pmod{p}$. This is also true for scalar multiplication: if α is sufficiently small then αZ_1 is consistent with $\alpha \cdot P_1 \pmod{p}$. However, the assumption that the digits are small is crucial for the homomorphisms to work, and jumping ahead, this will be the source of the gap in the proof.

⁶ It is worth mentioning that [21] also present a variant of their scheme for multivariate polynomials. This variant is somewhat different from the one described here and is obtained via a reduction to the univariate case.

Evaluation Proofs. Suppose that the committer wants to prove that $P(\zeta) = \gamma$, for some $\zeta = (\zeta_1, \dots, \zeta_n) \in \mathbb{F}^n$ and $\gamma \in \mathbb{F}$. More precisely, we will show an interactive protocol that is a *proof of knowledge* of an integer Z that is consistent with a polynomial P such that $C = g^Z$ and $P(\zeta) = \gamma$.

Let $P_0, P_1 : \mathbb{F}^{n-1} \rightarrow \mathbb{F}$ be the $(n-1)$ -variate polynomials defined as $P_0(\cdot) = P(0, \cdot)$ and $P_1(\cdot) = P(1, \cdot)$. The prover first generates these two polynomials, and the corresponding commitments $c_0 = g^{\mathcal{Z}(P_0)}$ and $c_1 = g^{\mathcal{Z}(P_1)}$. Also, let $\gamma_0 = P_0(\zeta_2, \dots, \zeta_n)$ and $\gamma_1 = P_1(\zeta_2, \dots, \zeta_n)$. As its first message, the prover sends $(c_0, c_1, \gamma_0, \gamma_1)$. The verifier now checks that:

1. $\gamma = \zeta_1 \cdot \gamma_1 + (1 - \zeta_1) \cdot \gamma_0$. This equation should indeed hold since $P(\zeta) = \zeta_1 \cdot P_1(\zeta_2, \dots, \zeta_n) + (1 - \zeta_1) \cdot P_0(\zeta_2, \dots, \zeta_n)$.
2. The verifier also checks that $c_0 \cdot (c_1)^{q^{N/2}} = c$, where $N := 2^n$. This should hold since

$$c_0 \cdot (c_1)^{q^{N/2}} = g^{\mathcal{Z}(P_0)} \cdot g^{q^{N/2} \cdot \mathcal{Z}(P_1)} = g^{\mathcal{Z}(P_0) + q^{N/2} \cdot \mathcal{Z}(P_1)} = g^{\mathcal{Z}(P)},$$

where the last equality follows from the fact that

$$\begin{aligned} \mathcal{Z}(P_0) + q^{N/2} \cdot \mathcal{Z}(P_1) &= \sum_{\mathbf{b} \in \{0,1\}^{n-1}} q^{\mathbf{b}} \cdot P_0(\mathbf{b}) + q^{N/2} \cdot \sum_{\mathbf{b} \in \{0,1\}^{n-1}} q^{\mathbf{b}} \cdot P_1(\mathbf{b}) \\ &= \sum_{\mathbf{b} \in \{0,1\}^n} q^{\mathbf{b}} \cdot P(\mathbf{b}) = \mathcal{Z}(P), \end{aligned}$$

where the arithmetic is over the integers and we leverage the homomorphic properties of the commitment. Note that actually computing the value $(c_1)^{q^{N/2}}$ is too expensive for the verifier.⁷ Thus, rather than computing it directly, this value is supplied by the prover who then proves its correctness using Wesolowski's [46] proof of exponentiation protocol.

Observe that we have replaced the single claim that we had about the tuple (c, ζ, γ) with two separate claims (c_0, ζ', γ_0) and (c_1, ζ', γ_1) , where $\zeta' = (\zeta_2, \dots, \zeta_n)$, on $(n-1)$ -variate polynomials so that if the original claim were true then the two resulting claims are true, whereas if the original claim is false then intuitively, at least one of the new claims is false.

Since we cannot afford to recurse on both claims, the next idea is to combine them into a single claim, using a random linear combination. In more detail, the verifier chooses a random coefficient⁸ $\alpha \in \mathbb{F}$ and sends this coefficient to the prover. Consider now a new commitment

$$c' = c_0 \cdot (c_1)^\alpha = g^{\mathcal{Z}(P_0) + \alpha \cdot \mathcal{Z}(P_1)}. \quad (1)$$

⁷ Computing this value directly by exponentiation takes time roughly $N = 2^n$ (using the standard repeated squaring trick) whereas we seek $\text{poly}(n)$ time verification. Note that since the group's order is not known, one cannot first compute $q^{N/2}$ modulo the group order, and only then exponentiate.

⁸ Looking ahead, it actually makes more sense to choose α from $\{0, \dots, 2^\lambda - 1\}$ where λ is a statistical security parameter (independent of the field size). We ignore this here and simply follow the presentation in [21].

At first glance, c' looks like a commitment to the (multilinear) polynomial $P'(\cdot) = P_0(\cdot) + \alpha \cdot P_1(\cdot)$. This is not exactly true since the operations in the exponent in Eq. (1) are over the integers rather than over the field \mathbb{F}_p . Nevertheless, it is indeed the case that when interacting with the honest prover, $c' = g^Z$, for an integer Z that is *consistent* with P' . The verifier would therefore like to check that $c' = g^Z$ such that Z is consistent with a polynomial P' such that $P'(\zeta') = \gamma'$, where $\gamma' \equiv \gamma_0 + \alpha \cdot \gamma_1 \pmod{p}$.

The parties have therefore reduced the instance (c, ζ, γ) to (c', ζ', γ') , of smaller dimension (since the new instance corresponds to a polynomial on $n - 1$ variables). At the bottom of the recursion (i.e., when the number of variables is 0), the parties are in the following situation - both hold a commitment $C_0 \in \mathbb{G}$ and a value $\gamma_0 \in \mathbb{F}_p$ and the claim is that $C_0 = g^{Z_0}$ such that $Z_0 = \gamma_0 \pmod{p}$. This can be checked by having the prover send Z_0 and the verifier explicitly checking that this value is consistent with γ_0 (and with C_0).

Bounding the Blowup in Coefficients. Note that as the protocol progresses, the magnitude of the digits in the base q representation of the integers grows. However, this growth is bounded - in every iteration the main source of growth is multiplication by α and so the growth is bounded by roughly a factor of p per iteration. Thus, by setting $q \gg p^n$ we ensure the growth of the coefficients does not break the homomorphism as the protocol progresses. This suffices for completeness. For soundness (or rather knowledge soundness), we actually need a larger bound on q and have the verifier check in the base of the recursion that $Z_0 \leq p^n$. Loosely speaking, this is done so that a cheating prover cannot use integers with large digits to violate the homomorphism.

2.2 A Gap in the Proof

We need to show that the above scheme is an argument-of-knowledge.⁹ Loosely speaking this means that for every polynomial-time prover strategy \mathcal{P} there exists a polynomial-time extractor \mathcal{E} so that for every input (c, ζ, γ) , if \mathcal{P} convinces V to accept with non-negligible probability, then $\mathcal{E}^{\mathcal{P}}$ outputs an integer Z such that $g = c^Z$ and Z is consistent with a polynomial P such that $P(\zeta) = \gamma$.

The extractor works recursively. Let us therefore assume that we have an extractor for the $(n - 1)$ -variate case and attempt to construct an extractor for the n -variate case. Thus, we are given a commitment c , a point $\zeta \in \mathbb{F}^n$ a value $\gamma \in \mathbb{F}$ and a prover that convinces the verifier to accept with non-negligible probability. For sake of this overview however, let us pretend that the prover succeeds with probability close to 1.

The high-level idea for extraction is as follows. First, let the prover send its first message which is $(c_0, c_1, \gamma_0, \gamma_1)$. At this point our extractor continues the

⁹ We note that [21] only aim to show that the protocol is an *argument* of knowledge (and this is inherent to their approach). Jumping ahead we mention that the evaluation proof in our variant of DARK will actually be a *proof* of knowledge (i.e., extraction is guaranteed even wrt computationally unbounded provers).

interaction with two uniformly random choices of α for the verifier, which we denote by $\hat{\alpha}$ and $\tilde{\alpha}$. This defines two claim triplets: $(\hat{c}, \hat{\zeta}', \hat{\gamma})$ and $(\tilde{c}, \tilde{\zeta}', \tilde{\gamma})$, where:

$$\begin{aligned} \hat{c} &= c_0 \cdot (c_1)^{\hat{\alpha}} & \tilde{c} &= c_0 \cdot (c_1)^{\tilde{\alpha}} \\ \hat{\gamma} &\equiv \gamma_0 + \hat{\alpha} \cdot \gamma_1 \pmod{p} & \tilde{\gamma} &\equiv \gamma_0 + \tilde{\alpha} \cdot \gamma_1 \pmod{p}. \end{aligned}$$

Since these two claims correspond to the $(n - 1)$ -variate case, we can now recursively run our extractor (twice) to obtain integers \hat{Z} and \tilde{Z} that are consistent with the respective claims. Namely, \hat{Z} (resp., \tilde{Z}) is consistent with a polynomial \hat{P} (resp., \tilde{P}) such that $\hat{P}(\hat{\zeta}') = \hat{\gamma}$ (resp., $\tilde{P}(\tilde{\zeta}') = \tilde{\gamma}$), and $g^{\hat{Z}} = \hat{c}$ (resp., $g^{\tilde{Z}} = \tilde{c}$).

Consider the following linear-system, over the rationals, with unknowns Z_0 and Z_1 .

$$\hat{Z} = Z_0 + \hat{\alpha} \cdot Z_1 \qquad \tilde{Z} = Z_0 + \tilde{\alpha} \cdot Z_1$$

Note that since $\hat{\alpha}$ and $\tilde{\alpha}$ are random, with overwhelming probability this system has a (unique) solution over the rationals:

$$Z_0 = \frac{\hat{\alpha} \cdot \tilde{Z} - \tilde{\alpha} \cdot \hat{Z}}{\hat{\alpha} - \tilde{\alpha}} \qquad Z_1 = \frac{\hat{Z} - \tilde{Z}}{\hat{\alpha} - \tilde{\alpha}} \quad (2)$$

An immediate difficulty that arises is that this solution may not be integral (i.e., Z_0 and Z_1 are not integers). However, Bünz et al. show that finding a fractional solution violates their hardness assumptions. Thus, (under the foregoing assumptions) we can safely assume that Z_0 and Z_1 are integers.

At this point we would like to combine Z_0 and Z_1 into $Z = Z_0 + q^{2^{n-1}} Z_1$, which serves as a valid output for the extractor. A question that arises however is whether Z_0 and Z_1 have bounded digits in their base q representation. This is crucial since, as discussed above, if the digits are large the homomorphism breaks. Bünz et al. claim that it is indeed the case that Z_0 and Z_1 have small coefficients by observing that both the numerators and denominators in Eq. (2) consist of relatively small integers and so their quotient is small. While the claim that the quotient itself is small is indeed valid, it does not necessarily mean that *the base q representation of the quotient has small digits*. Indeed, as demonstrated by the following example, this is not necessarily true and is the source of the gap in the DARK extraction procedure.

Example 2.1. Suppose that q is odd and consider the integers $a = q + 1$ and $b = 2$ (in case q is even a similar example with $a = q$ and $b = 2$ works). Note that the base q representation of both only has small digits. However, a/b has a digit of magnitude $(q + 1)/2$. Using such large digits breaks the homomorphism within a couple of steps.

We refer the reader to Lemma 8 in the full version of DARK [22] for the exact location of the gap in the proof. Specifically, in the third paragraph in that proof, it is claimed that $f_L(X)$ has small entries by the triangle inequality, but this does not account for the division by Δ_α in the definition of f_L . This division can entirely break the claimed bounds on the base q representation of f_L .

2.3 Resolving the Gap

Unfortunately, we do not know how to resolve the gap in the extraction procedure of [21]. Rather, we show how to modify the scheme and construct an extractor for our modified scheme.

As our first step, for a reason that will be made clear momentarily, rather than handling a single claim (c, ζ, γ) , we construct an interactive proof that handles a bundle of λ claims $\{(c_i, \zeta, \gamma_i)\}_{i \in [\lambda]}$, using the same evaluation point $\zeta \in \mathbb{F}^n$, and where λ is an auxiliary statistical security parameter. These λ claims do not have to be distinct, so to solve the original problem (c, ζ, γ) we can simply consider λ copies of it. Thus, our goal is to construct a proof-of-knowledge of integers Z_1, \dots, Z_λ that are consistent, respectively, with polynomials P_1, \dots, P_λ so that $P_i(\zeta) = \gamma_i$, for every $i \in [\lambda]$.

We follow the divide and conquer approach of [21]. Namely, using a similar type of interaction we split each one of the λ claims (c_i, ζ, γ_i) into two claims each on an $(n-1)$ -variate polynomial. At this point, we have, overall, 2λ claims on $(n-1)$ -variate polynomials and we would like to reduce these to just λ claims so that we can recurse. Denote this set of claims by $\{(c'_i, \zeta', \gamma'_i)\}_{i \in [2\lambda]}$ (note that the indexing intentionally ignores the source for each one of these claims).

Let us first describe how we generate a single claim from these 2λ claims. The verifier chooses a random subset $S \subseteq [2\lambda]$ and sends S to the prover. Consider now the new claim $(\bar{c}, \zeta', \bar{\gamma})$, where $\bar{c} = \prod_{i \in S} c'_i$ and $\bar{\gamma} \equiv \sum_{i \in S} \gamma'_i \pmod{p}$. If the original claims were true then with probability 1 the new claim is true, whereas, *intuitively*, if at least one of the original claims was false then with probability $1/2$ the new claim is false¹⁰. We therefore repeat this process λ times to derive λ claims so that if one of the original claims was false, then, with all but $2^{-\lambda}$ probability, one of the new claims will be false.

To actually make this argument work, we need to construct an extractor. As suggested above, the extractor can rewind the computation a constant r number of times to deduce a linear-system, analogous to that of Sect. 2.2, but now with $r \cdot \lambda$ equations and λ variables, where the coefficients are uniformly random 0/1 values.

Similarly to the situation in Sect. 2.2, it is clear that this linear-system is full rank (over the rationals) but it is not a priori clear that the solution is integral, nor that its base q representation has small digits. Nevertheless, we show that for *random* Boolean matrices this is indeed the case. This fact, which turns out to be non-trivial to prove, is summarized in the following lemma:

Lemma 2.2 (Informally Stated, see [12]). *If \mathbf{A} is uniformly random in $\{0, 1\}^{\lambda \times r \cdot \lambda}$ for $r \geq 5$, then with all but $2^{-\Omega(\lambda)}$ probability \mathbf{A} has a right-inverse $\mathbf{B} \in \mathbb{Z}^{r \cdot \lambda \times \lambda}$. Moreover, the inverse matrix \mathbf{B} can be found in $\text{poly}(\lambda)$ time and its entries have bit length at most $\text{poly}(\lambda)$.*

¹⁰ This is not actually precise since there are many polynomials that are consistent with the c'_i 's and so the claim could be true wrt some of these polynomials. This is dealt with formally by showing *knowledge soundness* (i.e., constructing an extractor).

Note that the fact that the inverse matrix \mathbf{B} of our linear-system has relatively small integral coefficients (independent of q) is crucial since it means that our solution is integral and has small digits in base q .

Our proof of Lemma 2.2 leverages ideas from the theory of integer lattices, see the full version for details [12]. Having found the desired solution to the specified linear-system, our extractor can proceed in the extraction similarly to the extraction in DARK. This concludes the high-level description of our resolution of the gap in the DARK scheme.

Remark 2.3. Note that our approach not only resolved the gap in the DARK extraction, but also unconditionally avoided the possibility of the linear-system having a non-integral solution. This simultaneously simplifies the definitions and proofs and lets us avoid an undesirable reliance on additional, poorly understood, cryptographic assumptions.

Remark 2.4. For sake of convenience, we used λ repetitions in our analysis. We remark however that the number of repetitions here is a *statistical security parameter*. Namely, it bounds even a computationally *unbounded* adversary's success probability by $2^{-\Omega(\lambda)}$. Thus, in practice it may be best to differentiate between this parameter and the cryptographic parameter that corresponds to the size of the group.

2.4 Small Space Implementation

Having resolved the gap in the security proof, we now turn our attention to implementing the polynomial commitment in small space.

When considering sublinear space algorithms it is important to specify how the input is given (since the algorithm cannot simply copy the input to its work tape, see [30] for a comprehensive discussion). For our context, the most natural choice is for our small space algorithms to be given *multi-pass streaming access* to the description of the multilinear polynomial. That is, the evaluations of the polynomial on the Boolean hypercube are written on the read-only input tape. The algorithm can process the input tape from left-to-right, or choose to reset the machine head to the beginning of the tape. The reason that this choice is natural is that when constructing our argument-system, we will need to apply this commitment to a transcript of a computation. Such a transcript can be generated in a (resettable) streaming manner by simply executing the computation.

With that in mind, let us first consider our commitment algorithm. Recall that we are given as input the stream of values $\{P(\mathbf{b})\}_{\mathbf{b} \in \{0,1\}^n}$, where $P : \mathbb{F}^n \rightarrow \mathbb{F}$ is a multilinear polynomial and we need to produce the commitment $g^{\mathcal{Z}(P)} = g^V$, where

$$V = \sum_{\mathbf{b} \in \{0,1\}^n} q^{\mathbf{b}} \cdot P(\mathbf{b}).$$

Note that we cannot compute V directly and then exponentiate. This is because even storing V requires 2^n bits. Rather, we will leverage the fact that V appears only in the exponent and compute g^V directly.

We do so by iterating through \mathbf{b} in lexicographic order while maintaining, as we go along, two variables C and D . We will maintain the invariant that at the start of the \mathbf{b} -th iteration $D = g^{q^{\mathbf{b}}}$ and $C = g^{V_{<\mathbf{b}}}$, where $V_{<\mathbf{b}} = \sum_{\mathbf{b}' < \mathbf{b}} q^{\mathbf{b}'} \cdot P(\mathbf{b}')$. To do so we:

- Initialize C as the group’s identity element and $D = g$.
- To update C and D from \mathbf{b} to $\mathbf{b} + 1$, we set $C \leftarrow C \cdot D^{P(\mathbf{b})}$ and $D \leftarrow D^q$ (using repeated squaring).

It is not hard to see that the invariant is indeed maintained. Given the value of D in the last iteration, it is easy to generate the commitment g^V .

Implementing the evaluation proofs is more subtle. The key challenge here is that throughout the recursion, the prover needs to deal with the intermediate polynomials that are defined throughout the recursion, but only has streaming access to the *original* base polynomial. Needless to say, we cannot afford to explicitly store the intermediate polynomials since this would introduce $2^{\Omega(n)}$ space usage.

Thus, we need, for sake of space efficiency, to open up the recursion and work directly with our original stream P . At first glance, one would hope that using the polynomial P we can emulate streaming access to the intermediate polynomials that we encounter. Unfortunately, we do not know how to do that. Rather, in order to commit or evaluate some intermediate polynomial Q , we show that as we process the base polynomial P , each value that we encounter, has some partial contribution to Q (with coefficients that depend on the verifier’s random challenges). The crucial observation is that both the commitment to Q and evaluation are *linear* and therefore commute. This means that we do not have to process the partial contributions of each entry of Q in sequence. Furthermore, we show that the coefficients of these entries in the linear combination can either be produced individually in small space (when evaluating the intermediate polynomials) or generated as a stream in small space (when producing commitments).

2.5 Generalizing Pietrzak’s Proof of Exponentiation Protocol

Our Proof of Exponentiation (PoE) protocol builds on Pietrzak’s PoE protocol [37]. We therefore start by recalling his protocol and then proceed to describe our improvement.

Pietrzak’s PoE protocol. Let \mathbb{G} be a group and $q \in \mathbb{Z}$. Recall that the prover wishes to prove that $y = x^{q^{2^t}}$ for some $x, y \in \mathbb{G}$ and $t \in \mathbb{N}$. As a shorthand, we will use $T = 2^t$ and denote the claim $y = x^{q^T}$ by the tuple (x, y, T) and refer to this as a claim of size T (because its validity can be easily checked by performing T repeated squarings). To proceed with the proof, the prover first sends a single group element $\mu = x^{q^{T/2}}$, which implicitly defines two sub-claims $(x, \mu, T/2)$ and $(\mu, y, T/2)$ of size $T/2$ each. Note that if (x, y, T) is true then both claims $(x, \mu, T/2)$ and $(\mu, y, T/2)$ must also be true: $y = x^{q^T}$ and $\mu = x^{q^{T/2}}$

implies that $y = \mu^{q^{T/2}}$. However, intuitively, if (x, y, T) is false then for any (even maliciously generated) μ , at least one of the sub-claims must be false. Instead of recursing on both subclaims, the prover combines the two subclaims into a single claim of size $T/2$. The new claim $(x', y', T/2)$ is computed by taking a, verifier specified, random linear combination of the two claims. That is,

$$x' = x^r \cdot \mu \quad \text{and} \quad y' = \mu^r \cdot y,$$

where $r \leftarrow \mathbb{Z}_{2\lambda}$ is sampled by the verifier. It is easy to see that if $(x, \mu, T/2)$ and $(\mu, y, T/2)$ are true then $(x', y', T/2)$ is also true, and Pietrzak (relying on QR_N^+ not having small order subgroups) shows that if one of $(x, \mu, T/2)$ or $(\mu, y, T/2)$ is false, then with overwhelming probability, over the choice of r , the claim $(x', y', T/2)$ is false. Now, the prover and verifier recurse on the $T/2$ -sized claim, halving the size every time and eventually ending up in the base case ($T = 1$) where the verifier just needs to check whether $y = x^q$ (which can be done in $\text{poly}(\lambda)$ time).

Our New PoE protocol. As mentioned above the main downside of Pietrzak’s protocol is that statistical soundness can only be proved for groups where small order subgroups do not exist. This difficulty arises since we are taking random linear combinations of *group* elements rather than *field* elements. In particular, if one of the group elements has small order, then the random linear combination does not have the desired effect.

Our approach for resolving this difficulty is inspired by our resolution for the gap in the DARK scheme (see Sects. 2.2 and 2.3). We will maintain more instances throughout the interaction and take random subset sums of all of these instances rather than random linear combinations of two instances. Interestingly, this simple idea gets us quite a bit of mileage - simplifying the analysis, improving the assumptions (e.g., in the case of class groups) and generalizing the result to general groups.

In more detail, rather than handling a single claim (x, y, T) , we will show a protocol for checking λ claims $\{(x_i, y_i, T)\}_{i \in [\lambda]}$ all sharing the same exponent parameter T . Note that the single claim case can be easily reduced to this more general setting by simply setting $x_1 = \dots = x_\lambda = x$ and $y_1 = \dots = y_\lambda = y$.

Analogous to Pietrzak’s protocol, our prover sends the sequence of values $\mu = (\mu_1, \dots, \mu_\lambda) \in \mathbb{G}^\lambda$ as its first message, where $\mu_i = x_i^{q^{T/2}}$, for every $i \in [\lambda]$. This decomposes the λ claims $\{(x_i, y_i, T)\}_{i \in [\lambda]}$ into two sets of λ claims each $\{(x_i, \mu_i, T/2)\}_{i \in [\lambda]}$ and $\{(\mu_i, y_i, T/2)\}_{i \in [\lambda]}$. It will be convenient however to think of these as a single set of claims $\{(z_i, w_i, T/2)\}_{i \in [2\lambda]}$. Note that if the original set of claims was not true, then no matter what values $\{\mu_i\}_{i \in [\lambda]}$ the prover sends, at least one of the claims in $\{(z_i, w_i, T/2)\}_{i \in [2\lambda]}$ must be false.

Reducing Claims via Subset Products. We show a simple and general method for reducing the number of claims. Let us first see how to produce a single new claim from these 2λ claims. The verifier chooses at random a set $S \subseteq [2\lambda]$ and sends S to the prover. Consider the claim $(z', w', T/2)$ where:

$$z' = \prod_{i \in S} z_i \quad \text{and} \quad w' = \prod_{i \in S} w_i.$$

Observe that if all the original claims were true (i.e., $w_i = z_i^{q^{T/2}}$, for every $i \in [\lambda]$) then $(z')^{q^{T/2}} = \prod_{i \in S} z_i^{q^{T/2}} = \prod_{i \in S} w_i = w'$ and so the resulting claim holds. On the other hand, if even just one of the original claims is false then:

$$\Pr_S \left[(z')^{q^{T/2}} = w' \right] = \Pr_S \left[\prod_{i \in S} z_i^{q^{T/2}} = \prod_{i \in S} w_i \right] = \Pr_S \left[\prod_{i \in S} u_i = \mathbf{1}_{\mathbb{G}} \right] \leq 1/2,$$

where $u_i = z_i^{q^{T/2}} \cdot w_i^{-1}$ for every $i \in [2\lambda]$, the group’s identity element is denoted by $\mathbf{1}$ and the inequality follows from the simple principle that a random subset product of a sequence of group elements, not all of which are equal to $\mathbf{1}$, is equal to $\mathbf{1}$ with probability at most $1/2$.

To get $2^{-\lambda}$ error probability, we simply repeat this process λ times to get a new sequence of λ claims, each of size $T/2$. We have thus reduced our λ size T claims to λ size $T/2$ claims. We can continue recursing as in Pietrzak’s protocol until $T = 1$ in which case the verifier can solve the problem by itself.

Remark 2.5. We remark that our technique introduces a factor of λ overhead in the communication complexity as compared to Pietrzak’s protocol. This is due to the fact that the prover has to send λ group elements per round.

Similarly to Remark 2.4, λ is a *statistical* (rather than computational) security parameter and relatively small values of λ may suffice. Moreover, we believe that it is possible to “interpolate” between our approach and that of [37] by considering the minimal sub-group size of \mathbb{G} and using coefficients of suitably larger magnitude in our choice of the random matrix.

3 Preliminaries

We let “ \circ ” denote the string concatenation operator and let ϵ denote the empty string; that is, for any string s it holds that $s = s \circ \epsilon = \epsilon \circ s$.

Let S be a finite, non-empty set. We let $x \leftarrow S$ denote sampling an element x uniformly at random from S . For any $N \in \mathbb{N}$, we let S^N denote the set of all sequences of length N containing elements of S , and note that $S^0 := \{\epsilon\}$. As usual, we make the convention that if $j > k$ then $\sum_{i=j}^k a_i = 0$ and $\prod_{i=j}^k a_i = 1$.

We let \mathbb{F}_p denote a finite field of prime cardinality p , and often use lower-case Greek letters to denote elements of \mathbb{F} , e.g., $\alpha \in \mathbb{F}$. We use boldface lowercase letters to denote binary vectors, e.g. $\mathbf{b} \in \{0, 1\}^n$. For bit strings $\mathbf{b} \in \{0, 1\}^n$, we naturally associate \mathbf{b} with integers in the set $\{0, 1, \dots, 2^n - 1\}$; i.e., $\mathbf{b} \equiv \sum_{i=1}^n b_i \cdot 2^{i-1}$. We assume that $\mathbf{b} = (b_n, \dots, b_1)$, where b_n is the most significant bit and b_1 is the least significant bit. For bit string $\mathbf{b} \in \{0, 1\}^n$ and $\sigma \in \{0, 1\}$ we let $\sigma\mathbf{b}$ (resp., $\mathbf{b}\sigma$) denote the string $(\sigma \circ \mathbf{b}) \in \{0, 1\}^{n+1}$ (resp., $(\mathbf{b} \circ \sigma) \in \{0, 1\}^{n+1}$). We use boldface lowercase to denote \mathbb{F} vectors, e.g., $\boldsymbol{\alpha} \in \mathbb{F}^n$. For $(\alpha_n, \dots, \alpha_1) =$

$\alpha \in \mathbb{F}^n$, we refer to α_n as the most significant field element and α_1 as the least significant field element. For two equal length vectors \mathbf{u}, \mathbf{v} , we let $\mathbf{u} \odot \mathbf{v}$ denote the coordinate-wise product of \mathbf{u} and \mathbf{v} . We let uppercase calligraphic letters denote sequences and let corresponding lowercase letters to denote its elements, e.g., $\mathcal{Y} = (y_{\mathbf{b}})_{\mathbf{b} \in \{0,1\}^n} \in \mathbb{F}^N$ is a sequence of N elements in \mathbb{F} . Often, for $\mathbf{b} \in \{0,1\}^n$, we let $\mathcal{Y}_{\mathbf{b}}$ denote the value $y_{\mathbf{b}}$.

We use upper case letters to denote matrices, e.g., $M \in \mathbb{Z}^{m \times n}$. For a matrix M of dimension $m \times n$, we let $M(i, *)$ and $M(*, j)$ denote the i^{th} row and j^{th} column of M , respectively. For row vector \mathbf{u} of length m and column vector \mathbf{v} of length n , we let $\mathbf{u} \cdot M$ and $M \cdot \mathbf{v}$ denote the standard matrix-vector product.

Non-standard Notation. We are also interested in matrix-vector “exponents”. Let \mathbb{G} be some group, $M \in \mathbb{Z}^{m \times n}$, $\mathbf{u} = (u_1, \dots, u_m) \in \mathbb{G}^{1 \times m}$, and $\mathbf{v} = (v_1, \dots, v_n)^\top \in \mathbb{G}^{n \times 1}$. We let $\mathbf{u} \star M$ and $M \star \mathbf{v}$ denote a matrix-vector exponent, defined as

$$(\mathbf{u} \star M)_j = \prod_{i=1}^m u_i^{M(i,j)} \qquad (M \star \mathbf{v})_{i'} = \prod_{j'=1}^n v_{j'}^{M(i',j')},$$

for every $j \in [n]$ and every $i' \in [m]$. Note that $\mathbf{u} \star M \in \mathbb{G}^{1 \times n}$ and $M \star \mathbf{v} \in \mathbb{G}^{m \times 1}$.

For vector $\mathbf{x} \in \mathbb{Z}^n$ and group element $g \in \mathbb{G}$, we abuse notation and define $g^{\mathbf{x}} := (g^{x_1}, \dots, g^{x_n})$. Finally, for $k \in \mathbb{Z}$ and vector $\mathbf{u} \in \mathbb{G}^n$, we let \mathbf{u}^k denote the vector $(u_1^k, \dots, u_n^k) \in \mathbb{G}^n$.

3.1 Multilinear Polynomials

An n -variate polynomial $f: \mathbb{F}^n \rightarrow \mathbb{F}$ is **multilinear** if the individual degree of each variable in f is at most 1.

Fact 3.1. *An multilinear polynomial $f: \mathbb{F}^n \rightarrow \mathbb{F}$ (over a finite field \mathbb{F}) is uniquely defined by its evaluations over the Boolean hypercube. Moreover, for every $\zeta \in \mathbb{F}^n$ it holds that $f(\zeta) = \sum_{\mathbf{b} \in \{0,1\}^n} f(\mathbf{b}) \cdot \prod_{i=1}^n \chi(b_i, \zeta_i)$, where $\chi(b, \zeta) = b \cdot \zeta + (1 - b) \cdot (1 - \zeta)$.*

As a short hand, we will often denote $\prod_{i=1}^n \chi(b_i, \zeta_i)$ by $\bar{\chi}(\mathbf{b}, \zeta)$ for $n = |\mathbf{b}| = |\zeta|$.

Notation for Multilinear Polynomials. Throughout this work, we represent n -variate, multilinear polynomials f by the N -sized sequence \mathcal{Y} containing evaluations of f over the Boolean hypercube. That is, $\mathcal{Y} := (f(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n}$, and denote the evaluation of the multilinear polynomial defined by \mathcal{Y} on ζ as $\text{ML}(\mathcal{Y}, \zeta) := \sum_{\mathbf{b}} \mathcal{Y}_{\mathbf{b}} \cdot \bar{\chi}(\mathbf{b}, \zeta)$. Furthermore, we also consider the evaluation of a multilinear polynomial defined by some integer sequence $\mathcal{Z} \in \mathbb{Z}^N$. For any $\zeta \in \mathbb{F}_p$ for prime p , we define $\text{ML}(\mathcal{Z}, \zeta) := \sum_{\mathbf{b}} (\mathcal{Z}_{\mathbf{b}} \bmod p) \cdot \bar{\chi}(\mathbf{b}, \zeta)$.

3.2 Groups of Hidden Order

We start by defining the notion of a *group sampler*.

Definition 3.2 (Group Sampler). *A PPT algorithm \mathcal{G} is a group sampler if for every $\lambda \in \mathbb{N}$, \mathcal{G} on input 1^λ samples a description¹¹ \mathbb{G} of a group of size at most 2^λ . As a shorthand, we denote this random process by $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$, and by $g \leftarrow \mathbb{G}$ denote the process of sampling a random group element from \mathbb{G} and assigning it to g .*

Furthermore, we say that \mathcal{G} is public-coin if the output of \mathcal{G} (i.e., the group description) is a uniformly random string.

Here, we will focus only on group samplers \mathcal{G} for which the Hidden Order Assumption holds, which, informally, requires that it be computationally hard to find (a multiple of) the order of a random group element of $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$.

Assumption 3.3 (Hidden Order Assumption). *The Hidden Order Assumption holds for \mathcal{G} if for every polynomial-size family of circuits $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$:*

$$\Pr [g^a = 1 \wedge a \neq 0 : \mathbb{G} \leftarrow \mathcal{G}(1^\lambda), g \leftarrow \mathbb{G}, a \leftarrow A_\lambda(\mathbb{G}, g)] \leq \text{negl}(\lambda). \quad (3)$$

Candidates for \mathcal{G} . In this work we consider two main candidates for \mathcal{G} where the Hidden Order Assumption is believed to hold:

1. **RSA group:** the multiplicative group \mathbb{Z}_N^* of integers modulo a product $N = P \cdot Q$ for large random primes P and Q . Here, the Hidden Order Assumption holds assuming the hardness of factoring N when it is a product of safe primes. This group can be sampled by choosing random primes and specifying their product. However, this is private-coin type generation and it is not clear how to generate the group in a public way (this corresponds to the well-studied problem of generating hard factoring instances using only public-coins).
2. **Class group:** the class group of an of imaginary quadratic order. Here, the Hidden Order Assumption (in fact, even much stronger assumptions) are believed to hold (see, e.g., [21, 46]). The main feature of such class groups is that there is a way to sample the group description using only public-coins. These are, to the best of our knowledge, the only known public-coin hidden order groups. We refer the reader to [21] for details.

3.3 Interactive Games and Proof Systems

Definition 3.4 (Merlin-Arthur Games). *Let r be a positive integer.*

An MA[2r] game (or just an MA game¹² if r is unspecified) is a tuple $\mathcal{G} = (1^r, 1^\ell, W)$, where $\ell \in \mathbb{Z}^+$ and $W \subseteq \{0, 1\}^$ is a set, called the win predicate, that*

¹¹ The group description includes a $\text{poly}(\lambda)$ description of the identity element, and $\text{poly}(\lambda)$ size circuits checking membership in the group, equality, performing the group operation and generating a random element in the group.

¹² MA stands for Merlin-Arthur proofs [3] (differing from Arthur-Merlin proofs in that the prover (Merlin) sends the first message).

is represented as a boolean circuit. The integer r is called the number of rounds of \mathcal{G} and $\{0, 1\}^\ell$ is called the challenge space.

If $\mathcal{G} = (1^r, 1^\ell, W)$ is an MA[2r] game and $P : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a function, then the value of \mathcal{G} with respect to P is denoted and defined as

$$v[P](\mathcal{G}) \stackrel{\text{def}}{=} \Pr_{\beta_1, \dots, \beta_r \leftarrow \{0, 1\}^\ell} [(\alpha_1, \beta_1, \dots, \alpha_r, \beta_r) \in W],$$

where each α_i denotes $P(\beta_1, \dots, \beta_{i-1})$. The value of \mathcal{G} , denoted $v(\mathcal{G})$, is $\sup_P \{v[P](\mathcal{G})\}$.

Definition 3.5 (Game Transcripts). If $\mathcal{G} = (1^r, 1^\ell, W)$ is an MA[2r] game, then a transcript for \mathcal{G} is a tuple $\tau = (\alpha_1, \beta_1, \dots, \alpha_r, \beta_r)$ with each $\beta_i \in \{0, 1\}^\ell$ and $\alpha_i \in \{0, 1\}^*$. If τ is contained in W , then it is said to be an accepting transcript for \mathcal{G} . If for a function $P : \{0, 1\}^* \rightarrow \{0, 1\}^*$, $\alpha_i = P(\beta_1, \dots, \beta_{i-1})$ for each $i \in [r]$, then τ is said to be consistent with P . If τ is both an accepting transcript for \mathcal{G} and consistent with P , we say simply that τ is an accepting transcript for (P, \mathcal{G}) .

Definition 3.6 (MA Verifiers). For a function $r : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ and a language \mathcal{L} , an MA[2r] verifier for \mathcal{L} is a polynomial-time algorithm V , where:

- V maps any string $x \in \{0, 1\}^*$ to an MA[2r(|x|)] game.¹³
- The completeness of V is a function $c : \mathbb{Z}^+ \rightarrow [0, 1]$, defined as

$$c(n) \stackrel{\text{def}}{=} \min_{x \in \mathcal{L} \cap \{0, 1\}^n} v(V(x)).$$

- The soundness error of V is a function $s : \mathbb{Z}^+ \rightarrow [0, 1]$, defined as

$$s(n) \stackrel{\text{def}}{=} \max_{x \in \{0, 1\}^n \setminus \mathcal{L}} v(V(x)).$$

Definition 3.7 (Witness-Extended Emulation (cf. [35])). An MA verifier V has (statistical) witness-extended $\epsilon(\cdot)$ -emulation with respect to a relation \mathcal{R} if there exists an expected polynomial-time oracle algorithm \mathcal{E} such that for all $P : \{0, 1\}^* \rightarrow \{0, 1\}^*$ and all $x \in \{0, 1\}^*$, if we sample $(\tau, w) \leftarrow \mathcal{E}^P(x)$, then:

- τ is distributed uniformly at random on the set of all possible transcripts between $V(x)$ and P .
- With all but $\epsilon(|x|)$ probability, if τ is an accepting transcript for $V(x)$ then $(x, w) \in \mathcal{R}$.

Definition 3.8. An MA verifier V has statistical witness-extended emulation with respect to a relation \mathcal{R} if it has statistical witness-extended ϵ -emulation (as per Definition 3.7) for some negligible function ϵ .

¹³ In particular, this definition implies there is a polynomial in n that bounds the length of any accepting transcript for $V(x)$ when $x \in \{0, 1\}^n$.

3.4 Multilinear Polynomial Commitment

Polynomial commitment schemes, introduced by Kate et al. [32] and generalized in [19, 21, 41, 45], are a cryptographic primitive that allows one to commit to a polynomial of bounded degree and later provably reveal evaluations of the committed polynomial. Since we consider only multilinear polynomials, we tailor our definition to them.

Convention. In defining the syntax of various protocols, we use the following convention for any list of arguments or returned tuple $(a, b, c; d, e)$ – variables listed before semicolon are known both to the prover and verifier whereas the ones after are only known to the prover. In this case, a, b, c are public whereas d, e are secret. In the absence of secret information the semicolon is omitted.

Definition 3.9 (Multilinear Polynomial Commitment Scheme). A multilinear polynomial commitment scheme is a tuple of protocols $(\text{Setup}, \text{Com}, \text{isValid}, \text{Eval})$ such that

1. $pp \leftarrow \text{Setup}(1^\lambda, p, 1^n)$: takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs public parameter pp that allows to support n -variate multilinear polynomials over $\mathbb{F} = \mathbb{F}_p$ for some prime p .
2. $(C; d) \leftarrow \text{Com}(pp, \mathcal{Y})$: takes as input public parameters pp and a description of a multilinear polynomial $\mathcal{Y} = (y_{\mathbf{b}})_{\mathbf{b} \in \{0,1\}^n}$ and outputs a commitment C and a (secret) decommitment d .
3. $b \leftarrow \text{isValid}(pp, C, \mathcal{Y}, d)$: takes as input pp , a commitment C , a description of the multilinear polynomial \mathcal{Y} and a decommitment d , and returns a decision bit $b \in \{0, 1\}$.
4. $\text{Eval}(pp, C, \zeta, \gamma; \mathcal{Y}, d)$: is a public-coin interactive proof system (P, V) for the relation:

$$\mathcal{R}_{\text{ml}} = \left\{ (pp, C, \zeta, \gamma; \mathcal{Y}, d) : \text{isValid}(pp, C, \mathcal{Y}, d) = 1 \wedge \gamma = \text{ML}(\mathcal{Y}, \zeta) \right\}, \quad (4)$$

where V is an MA verifier (as per Definition 3.6) where P is the honest strategy for V . Note that the verifier in this proof-system gets as input the public parameters pp , commitment C , evaluation point $\zeta \in \mathbb{F}^n$ and claimed evaluation $\gamma \in \mathbb{F}$, and the prover additionally receives the full description of the polynomial \mathcal{Y} and the decommitment d .

We require the following three properties from the scheme $(\text{Setup}, \text{Com}, \text{isValid}, \text{Eval})$:

1. **Perfect Correctness:** for all primes p , $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$ and all $\mathcal{Y} \in \mathbb{F}_p^{2^n}$ and $\zeta \in \mathbb{F}_p^n$,

$$\Pr \left[1 = \text{Eval}(pp, C, \zeta, \gamma; \mathcal{Y}, d) : \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda, p, 1^n), \\ (C; d) \leftarrow \text{Com}(pp, \mathcal{Y}), \gamma = \text{ML}(\mathcal{Y}, \zeta) \end{array} \right] = 1 .$$

2. **Computational Binding:** for every polynomial-sized family of circuits $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$ the following holds

$$\Pr \left[\begin{array}{l} (b_0 = b_1 = 1) \wedge (\mathcal{Y}_0 \neq \mathcal{Y}_1) : \\ \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda, p, 1^N) \\ (C, \mathcal{Y}_0, \mathcal{Y}_1, d_0, d_1) \leftarrow A_\lambda(pp) \\ b_0 \leftarrow \text{IsValid}(pp, C, \mathcal{Y}_0, d_0) \\ b_1 \leftarrow \text{IsValid}(pp, C, \mathcal{Y}_1, d_1) \end{array} \end{array} \right] \leq \text{negl}(\lambda) .$$

3. **Witness-Extended Emulation:** For $\text{Eval} = (P, V)$, V has (statistical) witness-extended emulation for the relation \mathcal{R}_{ml} (defined in Eq. (4)).

Remark 3.10. We note that our definition of polynomial commitment scheme is stronger than the ones used in the literature (see, e.g., [5, 7, 11, 21, 33, 34, 42, 45, 47, 48]), in that we require Eval to have *statistical* soundness (rather than computational). As a result we show soundness for every pair (x, pp) .

A key ingredient in our efficient argument-systems is polynomial commitments that can be generated in a time *and* space efficient way. We call such polynomial commitments *streamable*.

Definition 3.11 (Streamable Multilinear Polynomial Commitment Scheme). A streamable multilinear polynomial commitment scheme is a multilinear polynomial commitment scheme (as per Definition 3.9) with the following efficiency properties for n -variate multilinear polynomials over \mathbb{F}_p for some prime $p \leq 2^\lambda$:

1. The commitment output by Com is of size $n \cdot \text{poly}(\lambda)$, and assuming multi-pass streaming access to the description of the polynomial, the commitment can be implemented in time $2^n \cdot \text{poly}(n, \lambda)$ and space $\text{poly}(n, \lambda)$.
2. The communication complexity of the Eval protocol is $n \cdot \text{poly}(\lambda)$ and the receiver of Eval runs in time $\text{poly}(n, \lambda)$. Assuming multi-pass streaming access to the description of the polynomial, the committer of Eval can be implemented in time $2^n \cdot \text{poly}(n, \lambda)$ and space $\text{poly}(n, \lambda)$.

4 Our Results

In this section we formally state our main results. All analyses, protocols, and proofs are deferred to the full version [12] (unless otherwise stated).

Time- and Space-efficient Arguments. Our first main result is a time- and space-efficient public-coin zero-knowledge argument-system.

Theorem 4.1. Assume the existence of a group sampler for which the hidden order assumptions holds (see Assumption 3.3). Then, there exists a public-coin zero-knowledge argument-system for any \mathbf{NP} relation verifiable by a time T space S random access machine with the following complexity.

1. The protocol has perfect completeness and negligible soundness error.
2. The number of rounds is $O(\log T)$.
3. The communication complexity is $\text{poly}(\lambda, \log T)$.
4. The prover runs in time $T \cdot \text{poly}(\lambda, \log T)$ and space $S \cdot \text{poly}(\lambda, \log T)$.
5. The verifier runs in time $|x| \cdot \text{poly}(\lambda, \log T)$, for a given input $|x|$.

Theorem 4.1 relies on a new polynomial commitment scheme discussed next.

Streamable Polynomial Commitments. The core component of our time- and space-efficient arguments is a new polynomial commitment scheme for multilinear polynomials where the committer can be implemented in small space and verification is only poly-logarithmic.

Theorem 4.2. *Assume the existence of a group sampler for which the hidden order assumptions holds. Then, there exists a streamable multilinear polynomial commitment scheme (Setup, Com, isValid, Eval) (as per Definition 3.11) over finite field \mathbb{F} of prime-order p with the following efficiency guarantees:*

1. **Com** outputs a commitment of size $\text{poly}(\lambda)$ bits, runs in time $2^n \cdot \text{poly}(n, \lambda, \log(p))$ and space $n + O(\log(p)) + \text{poly}(\lambda)$, and uses a single pass over the stream;
2. **Eval** has $O(n)$ rounds and communication complexity $\text{poly}(n, \lambda, \log(p))$;
3. The committer of **Eval** runs in time $2^n \cdot \text{poly}(n, \lambda, \log(p))$ and space $n \cdot \text{poly}(\lambda, \log(p))$, and uses $O(n)$ passes over the stream; and
4. The receiver of **Eval** runs in time $\text{poly}(n, \lambda, \log(p))$.

We present our scheme in Sect. 5.

Proof-of-Exponentiation. Our polynomial commitment scheme relies on a new Proof-of-Exponentiation (PoE) protocol, which may be of independent interest.

For some group \mathbb{G} and base $q \in \mathbb{Z}$ consider the language

$$\mathcal{L}_{\mathbb{G},q} = \left\{ (x, y, t) \in \mathbb{G} \times \mathbb{G} \times \mathbb{N} : x^{q^{2^t}} = y \right\}. \quad (5)$$

Note that this problem can be solved in time roughly 2^t (by repeated squaring), but for some groups it is conjectured to not be solvable in significantly less time (even when leveraging parallelization). Indeed, an instantiation of this language using RSA groups underlies the original time-lock puzzle construction by Rivest, Shamir and Wagner [39]. This problem has also been used recently for constructing *verifiable delay functions* (VDFs). We show an extension of a recent protocol due to Pietrzak [37] that works for general groups.

Theorem 4.3. *Let \mathbb{G} be a group whose elements have $O(\log(|\mathbb{G}|))$ -bit descriptions, and whose group operations take time $\text{polylog}(|\mathbb{G}|)$, and let $q \in \mathbb{N}$. There exists a perfectly correct, statistically sound public-coin interactive-proof for $\mathcal{L}_{\mathbb{G},q}$ with the following efficiency properties for exponent parameter t :*

1. The communication complexity is $O(t\lambda^2 + t\lambda \log(|\mathbb{G}|))$ and there are t rounds.
2. The prover runs in time $2^t \cdot \text{poly}(\log(q), \log(|\mathbb{G}|), \lambda)$ and uses space $O(\lambda \cdot \log(|\mathbb{G}|) + \log(t) + \log(q) + \lambda^2)$
3. The verifier runs in time $t \cdot \text{poly}(\log(|\mathbb{G}|), \log(q), \lambda)$.

5 Multilinear Polynomial Commitment Scheme in Hidden Order Groups

We describe our commitment scheme ($\text{Setup}, \text{Com}, \text{isValid}, \text{Eval}$) for multilinear polynomials $f: \mathbb{F}^n \rightarrow \mathbb{F}$ over some field \mathbb{F} of prime-order p which is specified as an input to Setup . Throughout the section, we work with the description $\mathcal{Y} := (f(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n} \in \mathbb{F}^{2^n}$ of the multilinear polynomial f . First, in Sect. 5.1 we describe how to encode \mathcal{Y} as an integer. Then, in Sect. 5.2 we describe our polynomial commitment scheme.

5.1 Encoding Multilinear Polynomials as an Integer

One key portion of our scheme is encoding the sequence \mathcal{Y} , which defines our multilinear polynomial, as an integer. We do so by using a technique first introduced by [21]. Towards this, we first describe an encoding scheme for *integer* sequences. For any $N = 2^n$ and an odd integer $q \in \mathbb{N}$, let $\text{Enc}_q: \mathbb{Z}^N \rightarrow \mathbb{Z}$ be the function that encodes a sequence of integers $\mathcal{Z} \in \mathbb{Z}^N$ as¹⁴

$$\text{Enc}_q(\mathcal{Z}) := \sum_{\mathbf{b} \in \{0,1\}^n} q^{\mathbf{b}} \cdot \mathcal{Z}_{\mathbf{b}},$$

where $q^{\mathbf{b}}$ interprets \mathbf{b} (an n -bit string) as the naturally corresponding integer in the set $\{0, 1, \dots, N - 1\}$. To decode an integer $v \in \mathbb{Z}$, we output its base- q representation where, for convenience, the base- q digits of v are integers in the range $[-q/2, q/2)$. We refer to the decoding function as Dec_q .

Our Enc_q scheme has two homomorphic properties which we leverage to design our polynomial commitment. First, $\text{Enc}_q(\cdot)$ is a linear homomorphism over \mathbb{Z} ; that is, for any $\mathcal{Z}, \mathcal{Z}' \in \mathbb{Z}^N$ and $\alpha, \beta \in \mathbb{Z}$, it holds that $\alpha \cdot \text{Enc}_q(\mathcal{Z}) + \beta \cdot \text{Enc}_q(\mathcal{Z}') = \text{Enc}_q(\alpha \cdot \mathcal{Z} + \beta \cdot \mathcal{Z}')$. Second, $\text{Enc}_q(\cdot)$ satisfies a restricted form of multiplicative homomorphism; that is, for any $d \in \mathbb{N}$, we have $q^d \cdot \text{Enc}_q(\mathcal{Z}) = \text{Enc}_q((0^d, \mathcal{Z}))$.

Encoding Bounded Integer Sequences. In fact, looking ahead, we are interested in encoding only sequences of bounded integers. For some $B \in \mathbb{R}_{\geq 1}$, we let $\mathbb{Z}(B) := \{z \in \mathbb{Z}: -B \leq z < B\}$ be the set of integers whose absolute value is bounded by B . Then, to encode integer sequences in $\mathbb{Z}(B)^N$, we consider the restriction of Enc_q to the set $\mathbb{Z}(B)^N$. Notice that by definition, for any $\mathcal{Z} \in \mathbb{Z}(B)^N$, we have that $\text{Enc}_q(\mathcal{Z}) \in \mathbb{Z}(B \cdot (q^N - 1)/(q - 1))$. We remark that while Enc_q is not injective over all integer sequences (as integer sequences $(1+q, 0)$ and $(1, 1)$ both encode to the integer $1 + q$), the restriction of Enc_q to the set $\mathbb{Z}(q/2)^N$ is injective. We capture this in the following fact:

Fact 5.1 ([21, Fact 1]). *Let q be any odd integer and let $N \in \mathbb{N}$. For any $v \in \mathbb{Z}(q^N/2)$, there exists a unique sequence $\mathcal{Z} \in \mathbb{Z}(q/2)^N$ such that $v = \text{Enc}_q(\mathcal{Z})$. Furthermore, $\mathcal{Z} = \text{Dec}_q(v)$.*

¹⁴ This encoding is valid for sequences of arbitrary length, but we restrict to powers of two for convenience.

Proof. For any sequence $\mathcal{Z} \in \mathbb{Z}(q/2)^N$, by definition of Dec_q we observe that $\text{Dec}_q(\text{Enc}_q(\mathcal{Z})) = \mathcal{Z}$. This implies that (restriction of) Enc_q (to $\mathbb{Z}(q/2)^N$) is injective. Furthermore, the cardinality of sets $\mathbb{Z}(q^N/2)$ and $\mathbb{Z}(q/2)^N$ are equal. Therefore, for every $v \in \mathbb{Z}(q^N/2)$, $\text{Dec}_q(v)$ is the unique sequence in $\mathbb{Z}(q/2)^N$ that encodes to v .

Similar to Enc_q , Dec_q also satisfies some homomorphic properties: for integers z_1, z_2 , we have that $\text{Dec}_q(z_1 + z_2) = \text{Dec}_q(z_1) + \text{Dec}_q(z_2)$ as long as z_1, z_2 encode sequences whose elements are bounded by $q/4$. For our security proof, we will use the following more general statement, which we prove in the full version.

Claim 5.2. *Let $\ell, q, N \in \mathbb{N}$ such that q is odd, and let $B_1, B_2 \geq 1$ be such that $B_1 \cdot B_2 \leq q/(2\ell)$. Then, for every $\alpha_1, \dots, \alpha_\ell \in \mathbb{Z}(B_1)$, and integers $z_1, \dots, z_\ell \in \mathbb{Z}(q^N/2)$ such that $\text{Dec}_q(z_i) \in \mathbb{Z}(B_2)^N$,*

$$\text{Dec}_q\left(\sum_{i \in [\ell]} \alpha_i \cdot z_i\right) = \sum_{i \in [\ell]} \alpha_i \cdot \text{Dec}_q(z_i). \tag{6}$$

Remark 5.3. Looking ahead, the correctness of our extractor (to show security for our polynomial commitment scheme) relies crucially on Claim 5.2. The main issue with [21] is that their extractor relies on a variant of Claim 5.2 (formulated below) which is false. Lemma 8 in the full version [22] of [21] uses the following claim to argue correctness of the extracted integer decommitments f_L and f_R .

Claim 5.4 (False claim implicit in [22, Lemma 8]). *For $p, q, N \in \mathbb{N}$ such that $2 \leq p \leq q$ where q is odd. For every $\alpha \in \mathbb{Z}(p)$ and $z \in \mathbb{Z}(q^N/2)$ such that $\alpha \mid z$, $\text{Dec}_q(z/\alpha) = \text{Dec}_q(z)/\alpha$.*

We note that $z, z/\alpha \in \mathbb{Z}(q^N/2)$, by Fact 5.1 $\text{Dec}_q(z), \text{Dec}_q(z/\alpha) \in \mathbb{Z}(q/2)^N$. But, $\text{Dec}_q(z)/\alpha$ may not be an integer sequence. Counter-example: for $z = 1+q, \alpha = 2$, we have $\text{Dec}_q(z) = (1, 1)$ but $\text{Dec}_q(z)/2$ is not an integer sequence.

Encoding \mathcal{Y} . To encode $\mathcal{Y} \in \mathbb{F}^N$ where \mathbb{F} is a field of prime-order p , we first define a lifting function $\llbracket \cdot \rrbracket: \mathbb{F} \rightarrow \mathbb{Z}(p/2)$ in the natural way. That is, for any $\alpha \in \mathbb{F}$, we define $\llbracket \alpha \rrbracket$ to be the unique integer in $\mathbb{Z}(p/2)$ such that $\llbracket \alpha \rrbracket \equiv \alpha \pmod{p}$. We then define $\text{Enc}_q(\mathcal{Y})$ as $\text{Enc}_q(\mathcal{Y}) := \sum_{\mathbf{b} \in \{0,1\}^n} q^{\mathbf{b}} \cdot \llbracket \mathcal{Y} \mathbf{b} \rrbracket$.

5.2 Scheme

Our polynomial commitment scheme is parameterized by three components: (a) the encoding scheme $(\text{Enc}_q, \text{Dec}_q)$ defined in Sect. 5.1, (b) A group sampler \mathcal{G} for which the Hidden Order Assumption holds (see Sect. 3.2 for a discussion on candidates), and (c) a perfectly correct, statistically sound PoE protocol (we present one such protocol over arbitrary groups the full version). We now present all algorithms (Setup, Com, isValid, Eval) for the polynomial commitment scheme.

Setup($1^\lambda, p, 1^n$): On input security parameter 1^λ , a prime p , and the number of polynomial variables 1^n , the algorithm **Setup** samples group description $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$, samples $g \leftarrow \mathbb{G}$, sets $q := q(n, p, \lambda) \in \mathbb{N}$, and outputs public parameters $pp = (q, g, \mathbb{G})$. We require that q be odd such that $q > p \cdot 2^{n \cdot \text{poly}(\lambda)}$.

Com(pp, \mathcal{Y}): On input $pp = (q, g, \mathbb{G})$ output by **Setup** and sequence \mathcal{Y} , **Com** computes a commitment to the sequence \mathcal{Y} as $C = g^{\text{Enc}_q(\mathcal{Y})}$. The output of **Com** is the commitment C and secret decommitment $\mathcal{Z} = (\llbracket \mathcal{Y}_{\mathbf{b}} \rrbracket)_{\mathbf{b} \in \{0,1\}^n} \in \mathbb{Z}(p/2)^N$.

isValid($pp, C, \mathcal{Y}, \mathcal{Z}$): On inputs $pp = (q, g, \mathbb{G})$, C output by **Com**, committed sequence $\mathcal{Y} \in \mathbb{F}^N$ and decommitment $\mathcal{Z} \in \mathbb{Z}^N$ for $N = 2^n$, the algorithm **isValid** outputs a decision bit. **isValid** outputs 1 if and only if (1) $\mathcal{Z} \subseteq \mathbb{Z}(q/2)^N$; (2) $\mathcal{Y} \equiv \mathcal{Z} \pmod{p}$; and (3) $C = g^{\text{Enc}_q(\mathcal{Z})}$. Otherwise, **isValid** outputs 0.

Eval($pp, C, \zeta, \gamma; \mathcal{Y}, \mathcal{Z}$): On input $pp = (q, g, \mathbb{G})$, $C \in \mathbb{G}$, $\zeta \in \mathbb{F}^n$, $\gamma \in \mathbb{F}$, $\mathcal{Y} \in \mathbb{F}^N$ and $\mathcal{Z} \in \mathbb{Z}^N$ for $N = 2^n$, **Eval** is an interactive protocol (P, V) for the relation,

$$\mathcal{R}_{\text{ml}} = \left\{ (pp, C, \zeta, \gamma; \mathcal{Y}, \mathcal{Z}) : \text{isValid}(pp, C, \mathcal{Y}, \mathcal{Z}) = 1 \wedge \gamma = \text{ML}(\mathcal{Y}, \zeta) \right\}, \quad (7)$$

where on common input (pp, C, ζ, γ) , P tries to convince V that it knows a committed sequence $\mathcal{Y} \in \mathbb{F}^N$ and an integer sequence $\mathcal{Z} \in \mathbb{Z}^N$ such that **isValid**($pp, C, \mathcal{Y}, \mathcal{Z}$) = 1 and γ is the evaluation of the multilinear polynomial defined by \mathcal{Y} at evaluation point $\zeta = (\zeta_n, \dots, \zeta_1)$; that is, $\gamma \stackrel{?}{=} \text{ML}(\mathcal{Y}, \zeta)$. More specifically, both the committer and receiver in **Eval** first make λ many copies of the statement $(C, \zeta, \gamma; \mathcal{Z})$ as $(\mathbf{C}, \zeta, \gamma; Z)$, where $\mathbf{C} = (C, \dots, C) \in \mathbb{G}^\lambda$, $\gamma = (\gamma, \dots, \gamma) \in \mathbb{F}^\lambda$, and $Z \in \mathbb{Z}^{\lambda \times N}$ is a matrix such that $Z(i, \mathbf{b}) := Z_{\mathbf{b}}$ for every $i \in [\lambda]$ and $\mathbf{b} \in \{0,1\}^n$. The committer and receiver then run the subroutine **MultiEval**, presented in Algorithm 1.

MultiEval is a recursive protocol which given the statement $(\mathbf{C}, \zeta, \gamma; Z)$ proves that $\gamma_i = \text{ML}(Z(i, *), \zeta)$ and $C_i = \text{Com}(Z(i, *))$ for every $i \in [\lambda]$, where $Z(i, *) \in \mathbb{Z}^{1 \times N}$ is the i th row of Z . This is done via a divide and conquer approach. Let $P_i: \mathbb{F}^n \rightarrow \mathbb{F}$ be the multilinear polynomial defined by row i of matrix Z for every $i \in [\lambda]$. For presentation, we focus on the polynomial P_1 . To prove that $\gamma_1 = P_1(\zeta)$ and $C_1 = \text{Com}(P_1) = g^{\text{Enc}_q(P_1)}$, the committer first splits P_1 into its “left” and “right” halves, defined by $P_{1,L}(\cdot) = P_1(\cdot, 0)$ and $P_{1,R}(\cdot, 1)$. Then it computes evaluations of these polynomials at the point $\zeta' = (\zeta_n, \dots, \zeta_2)$ to obtain $\gamma_{1,L} = P_{1,L}(\zeta')$ and $\gamma_{1,R} = P_{1,R}(\zeta')$ (Line 5). Similarly, the committer also computes commitments $C_{1,L} = g^{\text{Enc}_q(P_{1,L})}$ and $C_{1,R} = g^{\text{Enc}_q(P_{1,R})}$ (Line 6). The claims $(\gamma_{1,L}, \gamma_{1,R})$ and $(C_{1,L}, C_{1,R})$ are then sent to the receiver. If indeed the committer defined $P_{1,L}$ and $P_{1,R}$ correctly, then $\gamma_1 = \gamma_{1,L} \cdot (1 - \zeta_1) + \gamma_{1,R} \cdot \zeta_1$ (Line 8) and $C_{1,L} \cdot C_{1,R}^{q^T} = C_1$ for $T = 2^{n-1}$. Since checking $C_{1,L} \cdot C_{1,R}^{q^T} = C_1$ directly is too costly to the receiver, the committer and prover run a *proof of exponent* protocol **PoE** to prove that equality holds (Line 9). The committer does simultaneously this for all polynomials P_i . The receiver then specifies random linear combinations $U \leftarrow \{0,1\}^{\lambda \times 2\lambda}$ (Line 10). The committer and receiver then obtain a set of λ new evaluations $\gamma'_i = \sum_{j \in [\lambda]} U(i, j) \cdot \gamma_{j,L} + U(i, 2j) \cdot \gamma_{j,R}$ and λ new

Algorithm 1: MultiEval($\mathbf{C}, k, \zeta, \gamma; Z$)**Input** : $\mathbf{C} \in \mathbb{G}^\lambda$, $k \in \mathbb{N}$, $\zeta \in \mathbb{F}^n$, $\gamma \in \mathbb{F}^\lambda$, and $Z \in \mathbb{Z}^{\lambda \times 2^{n-k}}$.**Output**: Accept or reject.

1 **if** $k = n$ **then**
2 P sends $Z \in \mathbb{Z}^\lambda$ to V .
3 V outputs accept if and only if $\|Z\|_\infty \leq p(2\lambda)^n$, $\gamma \equiv Z \pmod{p}$, and $\mathbf{C} = g^Z$.
4 **else**
5 P computes

$$\gamma_L = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} (Z(*, \mathbf{0b}) \pmod{p}) \cdot \prod_{j=1}^{n-k-1} \chi(b_j, \zeta_{j+k+1})$$

$$\gamma_R = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} (Z(*, \mathbf{1b}) \pmod{p}) \cdot \prod_{j=1}^{n-k-1} \chi(b_j, \zeta_{j+k+1})$$

6 P computes

$$\mathbf{C}_L = g^\ell \quad \text{where } \ell = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} q^{\mathbf{b}} \cdot Z(*, \mathbf{0b})$$

$$\mathbf{C}_R = g^{\mathbf{r}} \quad \text{where } \mathbf{r} = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} q^{\mathbf{b}} \cdot Z(*, \mathbf{1b})$$

7 P sends (γ_L, γ_R) and $(\mathbf{C}_L, \mathbf{C}_R)$ to V .
8 V checks $\gamma \stackrel{?}{=} \gamma_L \cdot (1 - \zeta_{k+1}) + \gamma_R \cdot \zeta_{k+1}$.
9 P and V run $\text{PoE}(\mathbf{C}_R, \mathbf{C}/\mathbf{C}_L, q, n-k-1, \lambda)$ which is a proof showing
 $\mathbf{C}_R(i)^{q^{2^{n-k-1}}} = \mathbf{C}(i)/\mathbf{C}_L(i)$ for every $i \in [\lambda]$. Here, \mathbf{C}/\mathbf{C}_L denotes
coordinate-wise division of the elements of \mathbf{C} by the elements of \mathbf{C}_L .
10 V samples $U = [U_L \| U_R] \leftarrow \{0,1\}^{\lambda \times 2\lambda}$ and sends U to P , where
 $U_L, U_R \in \{0,1\}^{\lambda \times \lambda}$.
11 P and V compute

$$\gamma' = U_L \cdot \gamma_L + U_R \cdot \gamma_R \quad \mathbf{C}' = (U_L \star \mathbf{C}_L) \odot (U_R \star \mathbf{C}_R)$$

12 For $Z_L, Z_R \in \{0,1\}^{\lambda \times 2^{n-k-1}}$ such that $Z = [Z_L \| Z_R]$, P computes

$$Z' = U_L \cdot Z_L + U_R \cdot Z_R.$$

13 **return** MultiEval($\mathbf{C}', k+1, \zeta, \gamma'; Z'$)

commitments $C'_i = \prod_{j \in [\lambda]} (C_{j,L})^{U(i,j)} \cdot (C_{j,R})^{U(i,2j)}$ (Line 11). This also defines new matrix $Z' = U_L \cdot Z_L + U_R \cdot Z_R$ (Line 12) for $U = [U_L \| U_R]$ and $Z = [Z_L \| Z_R]$. If the committer is honest, then the polynomial P'_1 defined by the row $Z'(1, *)$

satisfies $\gamma'_1 = P'_1(\zeta')$ and $C'_1 = g^{\text{Enc}_q(P'_1)}$ (and similarly for all other polynomials P'_i defined by row $Z'(i, *)$). The committer and receiver recurse via the above λ -to- 2λ -to- λ reduction until the matrix Z is a single column; at this point, Z is sent to the receiver. The receiver checks (Line 3) if the entries of Z are appropriately bounded, if the final vector $\gamma \equiv Z(\text{mod } p)$, and if $\mathbf{C} = g^Z = (g^{Z_1}, \dots, g^{Z_\lambda})$.

Remark 5.5. For simplicity of presentation, we let the (computational) security parameter λ_c given as input to **Setup** to be equal to the statistical security parameter λ_s given to **Eval**. However, they may be set differently: λ_c needs to be set so that 2^{λ_c} is larger than the running time of the adversary (generally, $\lambda_c = 2048$ for RSA groups to have security against 2^{80} time adversaries). However, λ_s needs to set so that the success probability of the adversary (we want to tolerate) is upperbounded by $2^{-\Omega(\lambda_s)}$, in fact, even relatively small values of λ_s would be sufficient for security, and offer qualitatively more efficient implementations.

Acknowledgments. Alexander R. Block was supported in part by NSF grant CCF-1910659. Pratik Soni was supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award. Ron Rothblum was supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grants No. 1262/18 and 2137/19), and grants from the Technion Hiroshi Fujiwara cyber security research center and Israel cyber directorate. Alon Rosen is supported in part by ISF grant No. 1399/17 and Project PROMETHEUS (Grant 780701).

References

1. Cash for RSA assumptions. <https://rsa.cash/rsa-assumptions/>
2. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426 (2019). <https://eprint.iacr.org/2019/426>
3. Babai, L., Moran, S.: Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. JCSS **36**(2), 254–276 (1988). [https://doi.org/10.1016/0022-0000\(88\)90028-1](https://doi.org/10.1016/0022-0000(88)90028-1). <https://www.sciencedirect.com/science/article/pii/0022000088900281>
4. Ben-Or, M., et al.: Everything provable is provable in zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 37–56. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_4
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast Reed-Solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, Prague, Czech Republic, 9–13 July 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>
6. Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S., Saraf, S.: Proximity gaps for Reed-Solomon codes. In: FOCS (2020)
7. Ben-Sasson, E., Goldberg, L., Kopparty, S., Saraf, S.: DEEP-FRI: sampling outside the box improves soundness. Cryptology ePrint Archive, Report 2019/336 (2019). <https://eprint.iacr.org/2019/336>

8. Ben-Sasson, E., Goldberg, L., Kopparty, S., Saraf, S.: DEEP-FRI: sampling outside the box improves soundness. In: Vidick, T. (ed.) ITCS (2020)
9. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 111–120. ACM Press, June 2013. <https://doi.org/10.1145/2488608.2488623>
10. Bitansky, N., Chiesa, A.: Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 255–272. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_16
11. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Public-coin zero-knowledge arguments with (almost) minimal time and space overheads. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 168–197. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_7
12. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Time- and space-efficient arguments from groups of unknown order. Cryptology ePrint Archive, Report 2021/358 (2021). <https://eprint.iacr.org/2021/358>
13. Blumberg, A.J., Thaler, J., Vu, V., Walfish, M.: Verifiable computation using multiple provers. Cryptology ePrint Archive, Report 2014/846 (2014). <http://eprint.iacr.org/2014/846>
14. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25
15. Boneh, D., Bünz, B., Fisch, B.: A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712 (2018). <https://eprint.iacr.org/2018/712>
16. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 561–586. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_20
17. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_12
18. Bowe, S., Grigg, J., Hopwood, D.: Halo: recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019). <https://eprint.iacr.org/2019/1021>
19. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press, May 2018. <https://doi.org/10.1109/SP.2018.00020>
20. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 1–18. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_1
21. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_24
22. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from DARK compilers. IACR Cryptology ePrint Archive 2019, 1229 (20200226:080105 (posted 26-Feb-2020 08:01:05 UTC)). <https://eprint.iacr.org/2019/1229>

23. Bünz, B., Fisch, B., Szepieniec, A.: Personal Communication (2021)
24. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_26
25. Chiesa, A., Ojha, D., Spooner, N.: FRACTAL: post-quantum and transparent recursive proofs from holography. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 769–793. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_27
26. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_8
27. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_17
28. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: SPARKs: succinct parallelizable arguments of knowledge. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 707–737. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_25
29. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
30. Goldreich, O.: Computational Complexity - A Conceptual Perspective. Cambridge University Press, New York (2008). <https://doi.org/10.1017/CBO9780511804106>
31. Holmgren, J., Rothblum, R.: Delegating computations with (almost) minimal time and space overhead. In: Thorup, M. (ed.) 59th FOCS, pp. 124–135. IEEE Computer Society Press, October 2018. <https://doi.org/10.1109/FOCS.2018.00021>
32. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11
33. Kattis, A., Panarin, K., Vlasov, A.: RedShift: transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400 (2019). <https://eprint.iacr.org/2019/1400>
34. Lee, J.: Dory: efficient, transparent arguments for generalised inner products and polynomial commitments. Cryptology ePrint Archive, Report 2020/1274 (2020). <https://eprint.iacr.org/2020/1274>
35. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptol.* **16**(3), 143–184 (2003). <https://doi.org/10.1007/s00145-002-0143-7>
36. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_13
37. Pietrzak, K.: Simple verifiable delay functions. In: Blum, A. (ed.) ITCS 2019. LIPIcs, vol. 124, pp. 60:1–60:15, January 2019. <https://doi.org/10.4230/LIPIcs.ITCS.2019.60>
38. Reingold, O., Rothblum, G.N., Rothblum, R.D.: Constant-round interactive proofs for delegating computation. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC, pp. 49–62. ACM Press, June 2016. <https://doi.org/10.1145/2897518.2897652>

39. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report, USA (1996)
40. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.* **25**(2), 252–271 (1996). <https://doi.org/10.1137/S0097539793255151>
41. Setty, S.: Spartan: efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) *CRYPTO 2020, Part III*. LNCS, vol. 12172, pp. 704–737. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_25
42. Setty, S., Lee, J.: Quarks: quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275 (2020). <https://eprint.iacr.org/2020/1275>
43. Tomescu, A.: Cryptographic assumptions in hidden-order groups. <https://alinush.github.io/2020/11/05/cryptographic-assumptions-in-hidden-order-groups.html>
44. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_1
45. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: *2018 IEEE Symposium on Security and Privacy*, pp. 926–943. IEEE Computer Society Press, May 2018. <https://doi.org/10.1109/SP.2018.00060>
46. Wesolowski, B.: Efficient verifiable delay functions. In: Ishai, Y., Rijmen, V. (eds.) *EUROCRYPT 2019, Part III*. LNCS, vol. 11478, pp. 379–407. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_13
47. Wijesekera, P., et al.: The feasibility of dynamically granted permissions: aligning mobile privacy with user preferences. In: *2017 IEEE Symposium on Security and Privacy*, pp. 1077–1093. IEEE Computer Society Press, May 2017. <https://doi.org/10.1109/SP.2017.51>
48. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: *2020 IEEE Symposium on Security and Privacy*, pp. 859–876. IEEE Computer Society Press, May 2020. <https://doi.org/10.1109/SP40000.2020.00052>

Encryption++



Broadcast Encryption with Size $N^{1/3}$ and More from k -Lin

Hoeteck Wee^{1,2(✉)}

¹ NTT Research, California, USA

² ENS, Paris, France

wee@di.ens.fr

Abstract. We present the first pairing-based ciphertext-policy attribute-based encryption (CP-ABE) scheme for the class of degree 3 polynomials with compact parameters: the public key, ciphertext and secret keys comprise $O(n)$ group elements, where n is input length for the function. As an immediate corollary, we obtain a pairing-based broadcast encryption scheme for N users with $O(N^{1/3})$ -sized parameters, breaking the long-standing \sqrt{N} barrier for pairing-based broadcast encryption. All of our constructions achieve adaptive security against unbounded collusions, and rely on the (bilateral) k -Lin assumption in prime-order bilinear groups.

1 Introduction

In this work, we study broadcast encryption [12] as well as attribute-based encryption schemes [5, 17, 23]. In ciphertext-policy attribute-based encryption (CP-ABE), ciphertexts ct are associated with a predicate f and a message m and keys sk with an attribute x , and decryption returns m when x satisfies f . Broadcast encryption is a special case of CP-ABE where the predicate is specified by a set $S \subseteq [N]$, and decryption returns m when $x \in S$. In both cases, we require security against unbounded collusions, so that an adversary that sees a ciphertext along with secret keys for an arbitrary number of attributes x_1, x_2, \dots learns nothing about m as long as none of these attributes satisfies f .

Broadcast encryption has been an active area of research since their introduction in the 1990s, where a major goal is to obtain schemes with short parameters, notably short ciphertexts ct and short public keys mpk . In a celebrated work from 2005, Boneh, Gentry and Waters (BGW) [6] presented a pairing-based broadcast encryption scheme with constant-size ciphertext (ignoring the contribution from the set S) and secret keys; however, the scheme has large public keys mpk which is linear in the total number of users N , and moreover, decryption requires access to mpk . To address these shortcomings, the authors also showed how to modify their scheme to achieve $O(\sqrt{N})$ -sized public keys, at the cost of a $O(\sqrt{N})$ -sized ciphertext. A series of follow-up works [7, 10, 15] showed how to achieve $O(\sqrt{N})$ -sized parameters (i.e., $|\text{mpk}| + |\text{ct}| + |\text{sk}| = O(\sqrt{N})$) under the standard k -Lin assumption, improving upon the q -type assumption used in BGW, while additionally strengthening the security guarantees from selective to adaptive security.

In a recent remarkable break-through, Agrawal and Yamada [2, 3] constructed a broadcast encryption scheme with $\text{poly}(\log N)$ -sized parameters from pairings

Scheme	$ \text{mpk} $	$ \text{ct} $	$ \text{sk} $	Assumption	Remark	Security
BGW05 [6]	$N^{1-\delta}$	N^δ	1^\dagger	q -type	$\delta \leq 1/2$	selective
[7, 10, 15]	\sqrt{N}	\sqrt{N}	1^\dagger	k -Lin, $k \geq 1$	$\delta \leq 1$	adaptive
	$N^{\max\{\delta, 1-\delta\}}$	N^δ	$N^{1-\delta}$			
this work	\sqrt{N}	\sqrt{N}	\sqrt{N}	bi- k -Lin*, $k \geq 2$	$\delta \leq 1/3$	adaptive
	$N^{1-2\delta}$	N^δ	$N^{1-2\delta}$			
	$N^{1/3}$	$N^{1/3}$	$N^{1/3}$			

Fig. 1. Comparison with prior pairing-based broadcast encryption schemes for N users, where the sizes refer to number of group elements, ignoring $O(1)$ factors. Note that $|\text{ct}|$ ignores the contribution from the set S , which is “public”. † In BGW05, decryption requires knowledge of mpk in addition to sk . Indeed, if we incorporate mpk into sk , then the secret key sizes matches those in the second row. * Here, bi- k -Lin (bilateral k -Lin) is a strengthening of k -Lin.

and LWE. Nonetheless, the following basic problem remains open since the work of BGW:

Can we build a broadcast encryption scheme with $o(\sqrt{N})$ -sized parameters (that is, $|\text{mpk}| + |\text{ct}| + |\text{sk}| = o(\sqrt{N})$) from (just) pairings?

Prior approaches for pairing-based broadcast encryption requires $|\text{ct}| \cdot \max\{|\text{sk}|, |\text{mpk}|\} = \Omega(N)$, which in turn implies a $\Omega(\sqrt{N})$ bound on the parameter size. Moreover, this is essentially optimal for a large class of approaches for pairing-based broadcast encryption [14], indicating that breaking the \sqrt{N} barrier would require substantially new ideas. As an aside –and an indication of our limited understanding of broadcast encryption with small parameters– we note that building a broadcast encryption scheme with $o(N)$ -sized ciphertext from just LWE is also an open problem.

1.1 Our Results

We present a pairing-based broadcast encryption scheme with $O(N^{1/3})$ -sized parameters, breaking the long-standing \sqrt{N} barrier. Our broadcast encryption scheme achieves adaptive security against unbounded collusions, and rely on the bilateral k -Lin assumption in prime-order bilinear groups. In addition, our construction offers a range of trade-offs between ciphertext and key sizes (see Fig. 1). We stress that prior to this work, it was not known how to achieve $o(\sqrt{N})$ -sized parameters with selective security even with q -type assumptions or generic bilinear groups.

More generally, we present a CP-ABE for degree 3 polynomials over $\{0, 1\}^n$ (and more generally, \mathbb{Z}_p^n) where the public key, ciphertext and secret keys comprise of $O(n)$ group elements; this scheme also achieves adaptive security against unbounded collusions under the bilateral k -Lin assumption. Our broadcast encryption scheme then follows as an immediate corollary, since we can encode set membership in $S \subseteq [N]$ as a degree 3 polynomial over $\{0, 1\}^{O(N^{1/3})}$. Prior to this work, CP-ABE schemes with $O(n)$ -sized parameters from pairings

Scheme	mpk	ct	sk	Assumption
inner product [10,18]	n^3	n^3	1	k -Lin, $k \geq 1$
	n^3	1	n^3	
degree 2 polynomials [22]	n^2	n^2	n	k -Lin, $k \geq 1$
	n^2	n	n^2	
this work	n	n	n	bi- k -Lin, $k \geq 2$

Fig. 2. Prior pairing-based CP-ABE for degree 3 polynomials $f : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$, where the sizes refer to number of group elements, ignoring $O(1)$ factors. These constructions follow from the fact that we can encode degree 3 polynomials as inner product of vectors of length $\mathbb{Z}_p^{n^3}$ or as degree 2 polynomials, and then combined with the appropriate ABE schemes in the literature. All of these schemes achieve adaptive security.

was only known for the class of degree 2 polynomials [22]. We refer to Fig. 2 for a summary of prior works on pairing-based CP-ABE for degree 3 polynomials.

The design of our schemes departs quite significantly from existing pairing-based ABE schemes, in that we exploit the power of “quadratic reconstruction”. This idea was previously used by Liu, Vaikuntanathan and Wee [22] to construct an information-theoretic, private-key analogue of broadcast construction –formally, conditional disclosure of secrets (CDS) for index– with $O(N^{1/3})$ -sized parameters. However, the scheme only works over fields of characteristic 2, which are incompatible with bilinear groups operations “in the exponent”. Instead, we provide new techniques for instantiating quadratic reconstruction that are inspired in part by recent works on functional encryption for degree 2 polynomials [13,21,26].

2 Technical Overview

We proceed to provide an overview of our constructions. We focus on our CP-ABE scheme for degree 3 polynomials over $\mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \mathbb{Z}_p^n$ given by

$$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \mapsto (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \cdot \mathbf{f}^\top$$

where $\mathbf{f} \in \mathbb{Z}_p^{n^3}$ is the coefficient vector. Throughout, we use boldface lower case to denote row vectors. In our CP-ABE scheme,

- encryption takes as input $\mathbf{f} \in \mathbb{Z}_p^{n^3}$ and a message M and outputs a ciphertext ct ;
- key generation takes as input $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathbb{Z}_p^n$ and outputs a key sk , and
- decryption takes as input ct, sk along with $\mathbf{f}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and outputs M whenever $(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)\mathbf{f}^\top \neq 0$.

We rely on an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ of prime order p where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We use $[\cdot]_1, [\cdot]_2, [\cdot]_T$ to denote component-wise exponentiations in respective groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. The k -Lin assumption in \mathbb{G}_1 asserts

that $([\mathbf{A}]_1, [\mathbf{sA}]_1) \approx_c ([\mathbf{A}]_1, [\mathbf{u}]_1)$ where $\mathbf{s} \leftarrow \mathbb{Z}_p^k, \mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \mathbf{u} \leftarrow \mathbb{Z}_p^{k+1}$. The bilateral k -Lin assumption (as used in this work, and slightly weaker than that used in [13, 26]) asserts that $([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{sA}]_2) \approx_c ([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{u}]_2)$, and is a strengthening of the k -Lin assumption in \mathbb{G}_2 . In symmetric bilinear groups, the bilateral k -Lin and the standard k -Lin assumption are equivalent. Note that 1-Lin = DDH/SXDH, and that bilateral 1-Lin is false, for the same reason DDH is false in symmetric bilinear groups. We will describe our construction based the k -Lin assumption and the bilateral k' -Lin assumption, and set $k = 1, k' = 2$ for optimal concrete efficiency.

Following [1, 26], we make extensive use of tensor products (cf. Sect. 3). This enables a more compact description of our schemes, and avoids triple summations to compute a degree 3 polynomial. Moreover, we will be replacing scalars with vectors as our schemes get increasingly complex, upon which some scalar-vector products translate naturally to a tensor product of two vectors, whereas some other ones translate to a vector-matrix product.

Roadmap. We will begin our overview by describing two candidate CP-ABE schemes for degree 3 polynomials. We refer to these schemes as “candidates” because we do not in fact prove “full fledged” security of these two schemes (though it does seem quite plausible that both schemes are secure in the generic group model).

- The first achieves

$$|\text{mpk}| = O(n^2), |\text{ct}| = O(n), |\text{sk}| = O(n)$$

In comparison, prior constructions based on degree 2 polynomials requires either $|\text{ct}| = O(n^2)$ or $|\text{sk}| = O(n^2)$ (cf. Fig 2).

- The second is a variant of the first with $|\text{mpk}| = O(n)$ and thus achieves $O(n)$ -sized parameters.

We then describe in Sect. 2.4 how to modify the second candidate to obtain our final CP-ABE scheme, which achieves $O(n)$ -sized parameters as well as adaptive security under the bi- k -Lin assumption.

2.1 CP-ABE for Degree 2 Polynomials

We begin with (a simplified variant of) the CP-ABE scheme in [22] for the class of degree 2 polynomials over $\mathbb{Z}_p^n \times \mathbb{Z}_p^n$ given by

$$(\mathbf{x}_1, \mathbf{x}_2) \mapsto (\mathbf{x}_1 \otimes \mathbf{x}_2) \cdot \mathbf{f}^\top$$

where $\mathbf{f} \in \mathbb{Z}_p^{n^2}$ is the coefficient vector and decryption is possible whenever $(\mathbf{x}_1 \otimes \mathbf{x}_2)\mathbf{f}^\top \neq 0$:

$$\begin{aligned} \text{mpk} &= [\alpha]_T, [\mathbf{w}_2]_1, [\mathbf{w}_1]_1, & \mathbf{w}_1 &\leftarrow \mathbb{Z}_p^n, \mathbf{w}_2 \leftarrow \mathbb{Z}_p^n, \alpha \leftarrow \mathbb{Z}_p \\ \text{ct} &= [s]_1, [((\mathbf{I}_{n_1} \otimes \mathbf{w}_2)\mathbf{f}^\top + \mathbf{w}_1^\top)s]_1, [\alpha s]_T \cdot M, s \leftarrow \mathbb{Z}_p & (1) \\ \text{sk} &= [r]_2, [\mathbf{x}_1 r \mathbf{w}_1^\top]_2, [\mathbf{x}_2 \alpha - r \mathbf{w}_2]_2, & r &\leftarrow \mathbb{Z}_p \end{aligned}$$

Note that the scheme achieves

$$|\text{mpk}| = O(n), |\text{ct}| = O(n), |\text{sk}| = O(n)$$

Decryption uses

$$\begin{aligned} (\mathbf{x}_1 \otimes \mathbf{x}_2) \mathbf{f}^\top \cdot \alpha s &= (\mathbf{x}_1 \otimes (\overbrace{(\mathbf{x}_2 \alpha - r \mathbf{w}_2)}^{\text{sk}} \overbrace{s}^{\text{ct}})) \mathbf{f}^\top \\ &+ \mathbf{x}_1 \overbrace{r}^{\text{sk}} \cdot \overbrace{((\mathbf{I}_{n_1} \otimes \mathbf{w}_2) \cdot \mathbf{f}^\top + \mathbf{w}_1^\top) s}^{\text{ct}} \\ &- \overbrace{\mathbf{x}_1 r \mathbf{w}_1^\top}^{\text{sk}} \cdot \overbrace{s}^{\text{ct}} \end{aligned} \quad (2)$$

Following the dual system encryption methodology [4, 19, 20, 24, 25], security boils down to showing that M is hidden given a single ciphertext-key pair. In particular, it suffices to show that if $(\mathbf{x}_1 \otimes \mathbf{x}_2) \mathbf{f}^\top = 0$, then α is hidden given

$$\begin{aligned} \widehat{\text{ct}} &= [(\mathbf{I}_n \otimes \mathbf{w}_2) \cdot \mathbf{f}^\top + \mathbf{w}_1^\top]_1, \\ \widehat{\text{sk}} &= [\mathbf{x}_1 \mathbf{w}_1^\top]_2, [\mathbf{x}_2 \alpha - \mathbf{w}_2]_2, \end{aligned} \quad (3)$$

where $\widehat{\text{ct}}, \widehat{\text{sk}}$ are derived from ct, sk by setting $r = s = 1$ and omitting $[\alpha s]_T$. Hiding of α then follows from

$$(\widehat{\text{ct}}, \widehat{\text{sk}}) \equiv (\widetilde{\mathbf{w}}_1^\top, ((\mathbf{x}_1 \otimes \widetilde{\mathbf{w}}_2) \mathbf{f}^\top + \mathbf{x}_1 \widetilde{\mathbf{w}}_1^\top - \overbrace{(\mathbf{x}_1 \otimes \mathbf{x}_2) \mathbf{f}^\top \cdot \alpha}^{=0}, \widetilde{\mathbf{w}}_2))$$

2.2 Our First Candidate CP-ABE

Next, we describe a candidate CP-ABE for degree 3 polynomials with parameter sizes

$$|\text{mpk}| = O(n^2), |\text{ct}| = O(n), |\text{sk}| = O(n)$$

To arrive at this scheme, we first replace \mathbf{x}_2 and \mathbf{w}_2 in (1) with $\mathbf{x}_2 \otimes \mathbf{x}_3$ and $\mathbf{w}_2 \otimes \mathbf{w}_3$ respectively, where $\mathbf{w}_3 \leftarrow \mathbb{Z}_p^n$. The ciphertext size remains unchanged, but the secret key size increases to $O(n^2)$ due to the term

$$(\mathbf{x}_2 \otimes \mathbf{x}_3) \alpha - r(\mathbf{w}_2 \otimes \mathbf{w}_3)$$

To achieve $|\text{sk}| = O(n)$, we will compute the above expression using

$$\mathbf{x}_2 \otimes \mathbf{x}_3 \alpha - r \mathbf{w}_2 \otimes \mathbf{w}_3 = \mathbf{x}_2 \otimes \overbrace{(\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3)}^{\text{sk}} - \overbrace{(\mathbf{x}_2 r_3 + r \mathbf{w}_2)}^{\text{sk}} \otimes \overbrace{\mathbf{w}_3}^{\text{ct}}$$

This yields the following scheme:

$$\begin{aligned} \text{mpk} &= [\alpha]_T & [\mathbf{w}_1]_1, [\mathbf{w}_3]_1, [\mathbf{w}_2 \otimes \mathbf{w}_3]_1, & \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \leftarrow \mathbb{Z}_p^n, \alpha \leftarrow \mathbb{Z}_p \\ \text{ct} &= [s]_1, [\alpha s]_T \cdot M, [((\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3) \cdot \mathbf{f}^\top + \mathbf{w}_1^\top) s]_1, [\mathbf{w}_3 s]_1, & s \leftarrow \mathbb{Z}_p, \\ \text{sk} &= [r_2]_2, & [\mathbf{x}_1 r_2 \mathbf{w}_1^\top]_2, [\mathbf{x}_2 r_3 + r_2 \mathbf{w}_2]_2, [\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3]_2, & r_2, r_3 \leftarrow \mathbb{Z}_p \end{aligned} \quad (4)$$

Here, we publish $[\mathbf{w}_2 \otimes \mathbf{w}_3]_1$ in mpk so that we can compute $[(\mathbf{w}_2 \otimes \mathbf{w}_3) s]_1$ in ct .

Compressing mpk. To get to a CP-ABE scheme with $O(n)$ -sized parameters, we will compress mpk in the previous scheme as follows: instead of having set-up pick \mathbf{w}_3 , the encryptor will sample a random \mathbf{w}_3 ; this eliminates $[\mathbf{w}_2 \otimes \mathbf{w}_3]_1$ in mpk and reduces mpk to $O(n)$ group elements. Next, we explain how this modification impacts ct and sk in (4):

- Given $[\mathbf{w}_2]_1, \mathbf{w}_3, s, \mathbf{f}$, it is easy to compute $[(\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 s) \cdot \mathbf{f}^\top]_1$ and thus $[((\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3) \cdot \mathbf{f}^\top + \mathbf{w}_1^\top) s]_1$ in ct.
- Now, key generation can no longer compute $[\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3]_2$, which was used to compute $[(\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3) s]_T$ during decryption. Instead, we will compute the latter using the equation

$$(\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3) s = \overbrace{(r_3 + r_2 v_0)}^{\text{sk}} \cdot \overbrace{\mathbf{w}_3 s}^{\text{ct}} + \overbrace{(\mathbf{x}_3 \alpha + r_2 \mathbf{v})}^{\text{sk}} \cdot \overbrace{s}^{\text{ct}} - r_2 \cdot \overbrace{(v_0 \mathbf{w}_3 + \mathbf{v})}^{\text{ct}}$$

where v_0, \mathbf{v} are chosen by the set-up algorithm.

Putting these modifications together, we obtain our next candidate.

2.3 Our Second Candidate CP-ABE

Here is our candidate CP-ABE scheme with $O(n)$ -sized parameters, where the terms not present in the previous scheme are shaded in gray:

$$\begin{aligned} \text{mpk} &= [\alpha]_T && [\mathbf{w}_2]_1, [\mathbf{w}_1]_1, && [\mathbf{v}]_1, [v_0]_1 \\ \text{ct} &= [s]_1, [\alpha s]_T \cdot M, [((\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3) \mathbf{f}^\top + \mathbf{w}_1^\top) s]_1, [\mathbf{w}_3 s]_1, && [(v_0 \mathbf{w}_3 + \mathbf{v}) s]_1, \\ \text{sk} &= [r_2]_2, && [\mathbf{x}_1 r_2 \mathbf{w}_1^\top]_2, [\mathbf{x}_2 r_3 + r_2 \mathbf{w}_2]_2, && [r_3 + r_2 v_0]_2, [\mathbf{x}_3 \alpha + r_2 \mathbf{v}]_2 \end{aligned} \quad (5)$$

$$\begin{aligned} \mathbf{w}_1, \mathbf{w}_2, \mathbf{v} &\leftarrow \mathbb{Z}_p^n, \alpha, v_0 \leftarrow \mathbb{Z}_p \\ \mathbf{w}_3 &\leftarrow \mathbb{Z}_p^n, s \leftarrow \mathbb{Z}_p, \\ r_3, r_2 &\leftarrow \mathbb{Z}_p \end{aligned}$$

The decryption algorithm on input $\text{ct} = ([s]_1, [\alpha s]_T \cdot M, [\mathbf{c}_1^\top]_1, [\mathbf{c}_2]_1, [\mathbf{c}_3]_1)$ and $\text{sk} = ([r_2]_2, [\mathbf{d}_1]_2, [\mathbf{d}_2]_2, [\mathbf{d}_3]_2, [\mathbf{d}_4]_2)$, computes $[(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \mathbf{f}^\top \cdot \alpha s]_T$ using

$$\begin{aligned} & \underbrace{(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes (\mathbf{d}_3 \mathbf{c}_2 + \mathbf{d}_4 s - r_2 \mathbf{c}_3))}_{= (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes (\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3)) s} \mathbf{f}^\top - \left(\underbrace{\mathbf{x}_1 \otimes (\mathbf{d}_2 (\mathbf{I}_n \otimes \mathbf{c}_2))}_{= (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes r_3 \mathbf{w}_3) s + (\mathbf{x}_1 \otimes r_2 \mathbf{w}_2 \otimes \mathbf{w}_3) s} \right) \mathbf{f}^\top \\ & + \underbrace{r_2 \mathbf{x}_1 \mathbf{c}_1^\top}_{= (\mathbf{x}_1 \otimes r_2 \mathbf{w}_2 \otimes \mathbf{w}_3) \mathbf{f}^\top s} - \underbrace{\mathbf{d}_1 s}_{(\text{iv})} \end{aligned}$$

where

$$\begin{aligned} \text{(i)} &= (r_3 + r_2 v_0) (\mathbf{w}_3 s) + (\mathbf{x}_3 \alpha + r_2 \mathbf{v}) s - r_2 (v_0 \mathbf{w}_3 + \mathbf{v}) s = (\mathbf{x}_3 \alpha + r_3 \mathbf{w}_3) s \\ \text{(ii)} &= (\mathbf{x}_2 r_3 + r_2 \mathbf{w}_2) \cdot (\mathbf{I}_n \otimes \mathbf{w}_3 s) = (\mathbf{x}_2 \otimes r_3 \mathbf{w}_3) s + (r_2 \mathbf{w}_2 \otimes \mathbf{w}_3) s \\ \text{(iii)} &= \mathbf{x}_1 r_2 ((\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3) \mathbf{f}^\top + \mathbf{w}_1^\top) s = (\mathbf{x}_1 \otimes r_2 \mathbf{w}_2 \otimes \mathbf{w}_3) \mathbf{f}^\top s + \mathbf{x}_1 r_2 \mathbf{w}_1^\top s \\ \text{(iv)} &= \mathbf{x}_1 r_2 \mathbf{w}_1^\top s \end{aligned}$$

Security warm-up. As before in Sect. 2.1, it suffices to show that α is computationally hidden given

$$\begin{aligned}\widehat{\text{ct}} &= [(\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3)\mathbf{f}^\top + \mathbf{w}_1^\top]_1, [\mathbf{w}_3]_1, [v_0\mathbf{w}_3 + \mathbf{v}]_1, \\ \widehat{\text{sk}} &= [\mathbf{x}_1\mathbf{w}_1^\top]_2, [\mathbf{x}_2r_3 + \mathbf{w}_2]_2, [r_3 + v_0]_2, [\mathbf{x}_3\alpha + \mathbf{v}]_2\end{aligned}\quad (6)$$

Here, we allow adaptive choices of \mathbf{f} and $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ subject to the constraint $(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)\mathbf{f}^\top = 0$. In this overview, we focus on the case \mathbf{f} is queried before $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

Step 1. We start by sampling random $\widetilde{\mathbf{w}}_1, \widetilde{\mathbf{v}}$ and programming

$$\widetilde{\mathbf{w}}_1^\top = (\mathbf{I}_n \otimes \mathbf{w}_2 \otimes \mathbf{w}_3)\mathbf{f}^\top + \mathbf{w}_1^\top, \quad \widetilde{\mathbf{v}} = v_0\mathbf{w}_3 + \mathbf{v}$$

We can then rewrite ct, sk as:

$$\begin{aligned}\text{ct} &= [\widetilde{\mathbf{w}}_1^\top]_1, [\mathbf{w}_3]_1, [\widetilde{\mathbf{v}}]_1 \\ \text{sk} &= [\mathbf{x}_1\widetilde{\mathbf{w}}_1^\top - (\mathbf{x}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3)\mathbf{f}^\top]_2, [\mathbf{x}_2r_3 + \mathbf{w}_2]_2, [r_3 + v_0]_2, [\mathbf{x}_3\alpha + \widetilde{\mathbf{v}} - v_0\mathbf{w}_3]_2\end{aligned}$$

Step 2. Next, we sample random $\widetilde{\mathbf{w}}_2, \widetilde{v}_0$ and program

$$\widetilde{\mathbf{w}}_2 = \mathbf{x}_2r_3 + \mathbf{w}_2, \quad \widetilde{v}_0 = r_3 + v_0$$

We can then rewrite sk as:

$$\begin{aligned}\text{sk} &= [\mathbf{x}_1\widetilde{\mathbf{w}}_1^\top - (\mathbf{x}_1 \otimes \widetilde{\mathbf{w}}_2 \otimes \mathbf{w}_3)\mathbf{f}^\top + (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes r_3\mathbf{w}_3)\mathbf{f}^\top]_2, \\ &[\widetilde{\mathbf{w}}_2]_2, [\widetilde{v}_0]_2, [\mathbf{x}_3\alpha + \widetilde{\mathbf{v}} - \widetilde{v}_0\mathbf{w}_3 + r_3\mathbf{w}_3]_2\end{aligned}$$

Step 3. At this point, all of the leakage on α comes from the following terms in ct, sk:

$$\begin{aligned}[\mathbf{w}_3]_1 \\ [(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes r_3\mathbf{w}_3)\mathbf{f}^\top]_2, [\mathbf{x}_3\alpha + r_3\mathbf{w}_3]_2\end{aligned}$$

If we can argue that $[r_3\mathbf{w}_3]_2$ is pseudorandom, then we have

$$\begin{aligned}\{[(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes r_3\mathbf{w}_3)\mathbf{f}^\top]_2, [\mathbf{x}_3\alpha + r_3\mathbf{w}_3]_2\} \approx_c [(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \widetilde{\mathbf{d}}_4)\mathbf{f}^\top \\ - \underbrace{(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)\mathbf{f}^\top}_{=0} \alpha]_2, [\widetilde{\mathbf{d}}_4]_2, \widetilde{\mathbf{d}}_4 \leftarrow \mathbb{Z}_p^n\end{aligned}$$

and then we are done. Unfortunately, $[r_3\mathbf{w}_3]_2$ is not pseudorandom given $[\mathbf{w}_3]_1$ for the same reason DDH is false in symmetric bilinear groups; however, an analogous statement does hold if we replace r_3, \mathbf{w}_3 with their k' -dimensional analogues ($k' \geq 2$). Concretely, the bilateral k' -Lin assumption tells us that $[r_3\mathbf{W}_3]_2$ is pseudorandom given $[\mathbf{W}_3]_1$, where $\mathbf{r}_3 \leftarrow \mathbb{Z}_p^{k'}$, $\mathbf{W}_3 \leftarrow \mathbb{Z}_p^{k' \times n}$.

Modifications. In addition to replacing r_3, \mathbf{w}_3 with $\mathbf{r}_3 \leftarrow \mathbb{Z}_p^{k'}, \mathbf{W}_3 \leftarrow \mathbb{Z}_p^{k' \times n}$,

- we replace $\mathbf{x}_2 r_3$ in sk with $\mathbf{x}_2 \otimes \mathbf{r}_3$, which in turns require increasing the width of \mathbf{w}_2 to $k'n$ (so that $\mathbf{x}_2 \otimes \mathbf{r}_3 + r_2 \mathbf{w}_2$ is well-defined);
- we replace $\mathbf{w}_2 \otimes \mathbf{w}_3 = \mathbf{w}_2(\mathbf{I}_n \otimes \mathbf{w}_3)$ in ct with $\mathbf{w}_2(\mathbf{I}_n \otimes \mathbf{W}_3)$;
- we replace v_0 with $\mathbf{v}_0 \in \mathbb{Z}_p^{k'}$.

This means that when we program $\tilde{\mathbf{w}}_2 = \mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{w}_2$ in Step 2, we have $\mathbf{w}_2(\mathbf{I}_n \otimes \mathbf{W}_3) = \tilde{\mathbf{w}}_2(\mathbf{I}_n \otimes \mathbf{W}_3) - \mathbf{x}_2 \otimes \mathbf{r}_3 \mathbf{W}_3$, upon which we could invoke the bi- k' -Lin assumption. The case \mathbf{f} is queried after $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ uses similar ideas, except we would instead rely on the k' -Lin assumption in \mathbb{G}_1 . Putting the modifications together, we arrive at the following variant of the scheme in (6):

$$\begin{aligned} \hat{\mathbf{ct}} &= [(\mathbf{I}_{n_1} \otimes \mathbf{w}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3))\mathbf{f}^\top + \mathbf{w}_1^\top]_1, [\mathbf{W}_3]_1, [\mathbf{v}_0 \mathbf{W}_3 + \mathbf{v}]_1, & \mathbf{W}_3 &\leftarrow \mathbb{Z}_p^{k' \times n} \\ \hat{\mathbf{sk}} &= [\mathbf{x}_1 \mathbf{w}_1^\top]_2, [\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{w}_2]_2, & [\mathbf{r}_3 + \mathbf{v}_0]_2, [\mathbf{x}_3 \alpha + \mathbf{v}]_2 & \mathbf{r}_3 \leftarrow \mathbb{Z}_p^{k'}, r_2 \leftarrow \mathbb{Z}_p \end{aligned} \quad (7)$$

In Lemma 1, we show that the above scheme hides α given $\hat{\mathbf{ct}}, \hat{\mathbf{sk}}$ for adaptive choices of \mathbf{f} and $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ subject to the constraint $(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)\mathbf{f}^\top = 0$. This holds under the k' -Lin assumption in \mathbb{G}_1 and the bi- k' -Lin assumption.

2.4 Our Final CP-ABE

We now describe how we arrive at our final CP-ABE for the class of degree 3 polynomials, which achieves adaptive security against unbounded collusions under the k -Lin assumption in $\mathbb{G}_1, \mathbb{G}_2$ and the bilateral k' -Lin assumption, where $k \geq 1, k' \geq 2$. Following the dual system encryption methodology and the ‘‘compiler’’ in [10], we sample $\mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1) \times k}, \mathbf{B} \leftarrow \mathbb{Z}_p^{k \times (k+1)}$ and make the following substitutions to the scheme in (5) combined with (7):

$$\begin{aligned} s &\mapsto \mathbf{A}\mathbf{s}^\top \in \mathbb{Z}_p^{1 \times (k+1)}, \quad \alpha \mapsto \mathbf{k} \in \mathbb{Z}_p^{k+1}, \quad r \mapsto \mathbf{r}\mathbf{B} \in \mathbb{Z}_p^{k+1} \\ \mathbf{w}_2 &\mapsto \mathbf{W}_2 \in \mathbb{Z}_p^{(k+1) \times (k+1)k'n}, \quad \mathbf{w}_1^\top \mapsto \mathbf{W}_1 \in \mathbb{Z}_p^{(k+1)n \times (k+1)}, \\ \mathbf{v} &\mapsto \mathbf{V} \in \mathbb{Z}_p^{(k+1) \times (k+1)n}, \quad \mathbf{v}_0 \mapsto \mathbf{V}_0 \in \mathbb{Z}_p^{(k+1) \times (k+1)k'} \end{aligned}$$

That is, we increase the width and heights of each of $\mathbf{w}_2, \mathbf{w}_1^\top, \mathbf{v}, \mathbf{v}_0$ by a multiplicative factor of $k+1$. We refer to Sect. 4.1 for a complete description of the scheme.

In the security proof, we rely on the following fact: for any $m, \ell \geq 1$, with probability $1 - 2/p$ over $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}, \mathbf{d} \leftarrow \mathbb{Z}_p^{k+1}$, the matrix

$$(\mathbf{I}_m \otimes \mathbf{d})\mathbf{M}(\mathbf{I}_\ell \otimes \mathbf{c}^\top) \in \mathbb{Z}_p^{m\ell}$$

is uniformly random given $\mathbf{M}(\mathbf{I}_\ell \otimes \mathbf{A}), (\mathbf{I}_m \otimes \mathbf{B})\mathbf{M}$, where $\mathbf{M} \leftarrow \mathbb{Z}_p^{(k+1)m \times (k+1)\ell}$. This was first observed in [10] for the special case $m = \ell = 1$. In our security reduction, we would then essentially ‘‘embed’’ $\mathbf{w}_2, \mathbf{w}_1^\top, \mathbf{v}, \mathbf{v}_0$ from the scheme in (7) into $\mathbf{d}\mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{c}^\top), (\mathbf{I}_{n_1} \otimes \mathbf{d})\mathbf{W}_1\mathbf{c}^\top, \mathbf{d}\mathbf{V}(\mathbf{I}_{n_3} \otimes \mathbf{c}^\top), \mathbf{d}\mathbf{V}_0(\mathbf{I}_{k'} \otimes \mathbf{c}^\top)$.

In the body of the paper, we consider a broader class of degree 3 polynomials over $\mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2} \times \mathbb{Z}_p^{n_3}$. By varying n_1, n_2, n_3 , we obtain trade-offs between ciphertext and key sizes as described in Fig. 1.

2.5 Discussion

We describe some additional related works as well as open problems.

The GKW lower bound. Gay, Kerendis and Wee showed a $N^{1/(d+1)}$ lower bound for information-theoretically secure conditional disclosure of secrets (CDS) protocols for broadcast encryption with degree d reconstruction [14]. The scheme in (3) constitutes such a CDS scheme with \sqrt{N} parameters and linear reconstruction, where the scheme in (6) constitutes a CDS scheme with computational security and $N^{1/3}$ parameters with quadratic reconstruction “in the exponent”. Given that quadratic reconstruction seems to be the best we can hope for with bilinear maps, beating the $N^{1/3}$ parameter size achieved in this work for pairing-based broadcast encryption would be a remarkable break-through.

poly(log N)-sized broadcast encryption. In 2014, Boneh, Waters and Zhandry constructed such a broadcast encryption scheme with poly(log N)-sized parameters assuming multi-linear maps [8]. As mentioned earlier, Agrawal and Yamada [2, 3] recently obtained the same result from pairings *and* LWE. Independently, Brakerski and Vaikuntathan [9] presented a “lattice-inspired” candidate broadcast encryption with poly(log N)-sized parameters, but they were unable to provide a reduction to LWE or any simple lattice assumption. These latter two works derived the broadcast encryption scheme as a special case of a more general result, namely CP-ABE for boolean formula/circuits over $\{0, 1\}^n$ with poly(n)-sized parameters.

$N^{1/3}$ -sized traitor-tracing. Zhandry [27] recently constructed the first pairing-based traitor-tracing scheme for N users with $O(N^{1/3})$ -sized parameters that is secure in the generic group model. While the work also constructed traitor-tracing schemes with broadcast, these additional schemes do not improve upon the state-of-the-art for broadcast encryption (see Table 1 in [27]), except for adding traitor-tracing capabilities. While Zhandry’s results did motivate us to revisit the LVW conjecture regarding a $O(N^{1/3})$ -sized broadcast encryption scheme, the techniques there-in appear to be largely unrelated to those developed in this work. In a way, broadcast encryption is harder than traitor-tracing in that we do have poly(log N)-sized traitor-tracing from just LWE [16], but not for broadcast encryption.

Open problems. We describe two open problems:

- Can we build a pairing-based CP-ABE for degree 2 polynomials with $|\text{mpk}| = O(n)$ and either $|\text{ct}| = O(1), |\text{sk}| = O(n)$ or $|\text{ct}| = O(n), |\text{sk}| = O(1)$? The former would imply a pairing-based broadcast encryption scheme for N users with $|\text{mpk}| = O(\sqrt{N}), |\text{ct}| = O(1), |\text{sk}| = O(\sqrt{N})$.

- Another important open problem is to build broadcast encryption with $O(\sqrt{N})$ -sized parameters, or CP-ABE for degree 2 polynomials with $O(n)$ -sized parameters from just LWE. All known approaches for LWE-based ABE has ciphertext size at least linear in the length of the attribute, which in the case of broadcast encryption means an $\Omega(N)$ -sized ciphertext. Much of the prior research efforts towards LWE-based CP-ABE has focused on the class of circuits, and perhaps it would be easier to make progress by focusing on the simple class of degree 2 polynomials.

Perspective. To conclude, our results provide the first indication that we could leverage techniques and insights from FE for degree 2 polynomials to achieve surprising asymptotic efficiency improvements in the broader setting of pairing-based ABE. We are optimistic that this connection could yield further (asymptotic) efficiency improvements in other pairing-based schemes, both within ABE and beyond.

3 Preliminaries

Notations. We denote by $s \leftarrow S$ the fact that s is picked uniformly at random from a finite set S . We use \approx_s to denote two distributions being statistically indistinguishable, and \approx_c to denote two distributions being computationally indistinguishable. We use lower case boldface to denote *row* vectors and upper case boldface to denote matrices. For any positive integer N , we use $[N]$ to denote $\{1, 2, \dots, N\}$.

Tensor product. The tensor product (Kronecker product) for matrices $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}^{\ell \times m}$, $\mathbf{B} \in \mathbb{Z}^{n \times p}$ is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B}, & \dots, & a_{1,m}\mathbf{B} \\ \dots, & \dots, & \dots \\ a_{\ell,1}\mathbf{B}, & \dots, & a_{\ell,m}\mathbf{B} \end{bmatrix} \in \mathbb{Z}^{\ell n \times mp}.$$

The mixed-product property for tensor product says that

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$$

A useful corollary of the mixed-product property says that for any pair of row vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^n$,

$$\begin{aligned} \mathbf{u} \otimes \mathbf{v} &= (\mathbf{u} \otimes 1)(\mathbf{I}_n \otimes \mathbf{v}) = (1 \otimes \mathbf{v})(\mathbf{u} \otimes \mathbf{I}_n) \\ &= \mathbf{u}(\mathbf{I}_n \otimes \mathbf{v}) = \mathbf{v}(\mathbf{u} \otimes \mathbf{I}_n) \end{aligned}$$

We adopt the convention that matrix multiplication takes precedence over tensor product, so that we can write $\mathbf{A} \otimes \mathbf{BC}$ to mean $\mathbf{A} \otimes (\mathbf{BC})$.

3.1 Prime-Order Bilinear Groups

A generator \mathcal{G} takes as input a security parameter 1^λ and outputs a description $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where p is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are cyclic groups of order p , and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear map. We require that the group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the bilinear map e are computable in deterministic polynomial time in λ . Let $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $g_T = e(g_1, g_2) \in \mathbb{G}_T$ be the respective generators. We employ the *implicit representation* of group elements: for a matrix \mathbf{M} over \mathbb{Z}_p , we define $[\mathbf{M}]_1 := g_1^{\mathbf{M}}, [\mathbf{M}]_2 := g_2^{\mathbf{M}}, [\mathbf{M}]_T := g_T^{\mathbf{M}}$, where exponentiation is carried out component-wise. Also, given $[\mathbf{A}]_1, [\mathbf{B}]_2$, we let $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$. We recall the matrix Diffie-Hellman (MDDH) assumption on \mathbb{G}_1 [11]:

Assumption 1 (MDDH $_{k,\ell}^d$ Assumption). *Let $k, \ell, d \in \mathbb{N}$. We say that the MDDH $_{k,\ell}^d$ assumption holds if for all PPT adversaries \mathcal{A} , the following advantage function is negligible in λ .*

$$\text{Adv}_{\mathcal{A}}^{\text{MDDH}_{k,\ell}^d}(\lambda) := \left| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, \boxed{[\mathbf{MS}]_1}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{U}]_1) = 1] \right|$$

where $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda), \mathbf{M} \leftarrow \mathbb{Z}_p^{\ell \times k}, \mathbf{S} \leftarrow \mathbb{Z}_p^{k \times d}$ and $\mathbf{U} \leftarrow \mathbb{Z}_p^{\ell \times d}$.

The MDDH assumption on \mathbb{G}_2 can be defined in an analogous way. Escala et al. [11] showed that

$$k\text{-Lin} \Rightarrow \text{MDDH}_{k,k+1}^1 \Rightarrow \text{MDDH}_{k,\ell}^d \quad \forall k, d \geq 1, \ell > k$$

with a tight security reduction. (In the setting where $\ell \leq k$, the MDDH $_{k,\ell}^d$ assumption holds unconditionally.)

The bilateral MDDH assumption is defined analogously with the advantage function:

$$\left| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, \boxed{[\mathbf{MS}]_1}, [\mathbf{M}]_2, \boxed{[\mathbf{MS}]_2}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{U}]_1, [\mathbf{M}]_2, [\mathbf{U}]_2) = 1] \right|$$

Note that the bilateral MDDH and bilateral k -Lin assumptions are false for $k = 1$. In this paper, we only require a weaker variant of the bilateral MDDH assumption, as defined with the advantage function:

$$\left| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{M}]_2, \boxed{[\mathbf{MS}]_2}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{M}]_2, [\mathbf{U}]_2) = 1] \right|$$

3.2 Attribute-Based Encryption

We define attribute-based encryption in the framework of key encapsulation. A attribute-based encryption scheme for a predicate $P(\cdot, \cdot)$ consists of four algorithms (Setup, Enc, KeyGen, Dec):

Setup($1^\lambda, \mathcal{X}, \mathcal{Y}$) \rightarrow (pp, mpk, msk). The setup algorithm gets as input the security parameter λ , the the predicate domains \mathcal{X}, \mathcal{Y} and outputs the public parameter mpk, and the master key msk.

$\text{Enc}(\text{mpk}, x) \rightarrow (\text{ct}, \kappa)$. The encryption algorithm gets as input mpk and $x \in \mathcal{X}$.

It outputs a ciphertext ct and a symmetric key $\text{kem} \in \{0, 1\}^\lambda$.

$\text{KeyGen}(\text{msk}, y) \rightarrow \text{sk}$. The key generation algorithm gets as input msk and $y \in \mathcal{Y}$.

It outputs a secret key sk .

$\text{Dec}(\text{sk}, y, \text{ct}, x) \rightarrow \kappa$. The decryption algorithm gets as input $\text{sk}, \text{ct}, x, y$ such that $\text{P}(x, y) = 1$. It outputs a symmetric key kem .

In our schemes, we would actually compute $\text{kem} \in \mathbb{G}_T$, which can then be hashed to $\{0, 1\}^\lambda$.

Correctness. We require that for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $\text{P}(x, y) = 1$,

$$\Pr[(\text{ct}, \text{kem}) \leftarrow \text{Enc}(\text{mpk}, x); \text{Dec}(\text{sk}, y, \text{ct}, x) = \text{kem}] = 1,$$

where the probability is taken over $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y})$ and the coins of Enc .

Security definition. For a stateful adversary \mathcal{A} , we define the advantage function

$$\text{Adv}_{\mathcal{A}}^{\text{ABE}}(\lambda) := \Pr \left[\begin{array}{l} (\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}); \\ x \leftarrow \mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot)}(\text{mpk}); \\ b = b' : b \leftarrow_{\text{R}} \{0, 1\}; \text{kem}_1 \leftarrow_{\text{R}} \{0, 1\}^\lambda \\ (\text{ct}, \text{kem}_0) \leftarrow \text{Enc}(\text{mpk}, x); \\ b' \leftarrow \mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot)}(\text{ct}, \text{kem}_b) \end{array} \right] - \frac{1}{2}$$

with the restriction that all queries y that \mathcal{A} makes to $\text{KeyGen}(\text{msk}, \cdot)$ satisfies $\text{P}(x, y) = 0$. An attribute-based encryption scheme is *adaptively secure* if for all PPT adversaries \mathcal{A} , the advantage $\text{Adv}_{\mathcal{A}}^{\text{ABE}}(\lambda)$ is a negligible function in λ .

CP-ABE for degree 3 polynomials. Here,

$$\mathcal{X} = \mathbb{Z}_p^{n_1 n_2 n_3}, \mathcal{Y} = \mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2} \times \mathbb{Z}_p^{n_3}$$

and

$$\text{P}(\mathbf{f}, (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)) = 1 \iff (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \cdot \mathbf{f}^T \neq 0$$

Broadcast Encryption. Here,

$$\mathcal{X} = \{0, 1\}^N, \mathcal{Y} = [N]$$

where we think of $\{0, 1\}^N$ as the power set of $[N]$ (i.e., set of all subsets of $[N]$), and

$$\text{P}(S, y) = 1 \iff y \in S$$

4 CP-ABE for Degree 3 Polynomials

In this section, we present an adaptively secure CP-ABE for degree 3 polynomials against unbounded collusions, under the k -Lin assumption in $\mathbb{G}_1, \mathbb{G}_2$ and the bilateral k' -Lin assumption, where $k \geq 1, k' \geq 2$. Our scheme achieves

$$\begin{aligned} |\text{mpk}| &= (k(k+1) + k(k+1)(n_1 + k'n_2 + n_3) + k')|\mathbb{G}_1| + |\mathbb{G}_T| \\ |\text{ct}| &= (k+1 + (k+1)n_1 + (k+1)k'n_3 + (k+1)n_3)|\mathbb{G}_1| \\ |\text{sk}| &= (2k+1 + (k+1)k'n_2 + (k+1)k' + (k+1)n_3)|\mathbb{G}_2| \end{aligned}$$

Setting $k = 1, k' = 2$, we obtain

$$\begin{aligned} |\text{mpk}| &= (2n_1 + 4n_2 + 2n_3 + 4)|\mathbb{G}_1| + |\mathbb{G}_T|, \quad |\text{ct}| = (2n_1 + 6n_3 + 2)|\mathbb{G}_1|, \\ |\text{sk}| &= (4n_2 + 2n_3 + 7)|\mathbb{G}_2| \end{aligned}$$

4.1 Our Scheme

– **Setup**($p, 1^{n_1}, 1^{n_2}, 1^{n_3}$): Run $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(p)$. Sample

$$\begin{aligned} \mathbf{A} &\leftarrow \mathbb{Z}_p^{(k+1) \times k}, \mathbf{k} \leftarrow \mathbb{Z}_p^{k+1}, \mathbf{W}_2 \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)k'n_2}, \mathbf{W}_1 \leftarrow \mathbb{Z}_p^{(k+1)n_1 \times (k+1)}, \\ \mathbf{V} &\leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)n_3}, \mathbf{V}_0 \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)k'}, \mathbf{B} \leftarrow \mathbb{Z}_p^{k \times (k+1)} \end{aligned}$$

For a matrix $\mathbf{M} \in \mathbb{Z}_p^{(k+1)m \times (k+1)\ell}$, we write $\overline{\mathbf{M}} := \mathbf{M}(\mathbf{I}_\ell \otimes \mathbf{A}) \in \mathbb{Z}_p^{(k+1)m \times k\ell}$. In particular, we have

$$\overline{\mathbf{k}} = \mathbf{k}\mathbf{A}, \quad \overline{\mathbf{W}}_2 = \mathbf{W}_2(\mathbf{I}_{k'n_2} \otimes \mathbf{A}), \quad \overline{\mathbf{W}}_1 = \mathbf{W}_1\mathbf{A}, \quad \overline{\mathbf{V}} = \mathbf{V}(\mathbf{I}_{n_3} \otimes \mathbf{A}), \quad \overline{\mathbf{V}}_0 = \mathbf{V}_0(\mathbf{I}_{k'} \otimes \mathbf{A})$$

Output

$$\text{mpk} = (\mathbb{G}, [\mathbf{A}]_1, [\overline{\mathbf{k}}]_T, [\overline{\mathbf{W}}_2]_1, [\overline{\mathbf{W}}_1]_1, [\overline{\mathbf{V}}]_1, [\overline{\mathbf{V}}_0]_1), \quad \text{msk} = (\mathbf{k}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{V}, \mathbf{V}_0, \mathbf{B})$$

– **Enc**(mpk, \mathbf{f}): Sample

$$\mathbf{s} \leftarrow \mathbb{Z}_p^k, \mathbf{W}_3 \leftarrow \mathbb{Z}_p^{k' \times n_3}$$

and output

$$\begin{aligned} \text{ct} &= \left(\underbrace{[\mathbf{A}\mathbf{s}^\top]_1}_{\mathbf{c}_0^\top}, \underbrace{[(\mathbf{I}_{n_1} \otimes (\overline{\mathbf{W}}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{s}^\top)))\mathbf{f}^\top + \overline{\mathbf{W}}_1\mathbf{s}^\top]_1}_{\mathbf{c}_1^\top} \right), \\ &\underbrace{[\mathbf{W}_3 \otimes \mathbf{A}\mathbf{s}^\top]_1}_{\mathbf{C}_2}, \underbrace{[\overline{\mathbf{V}}_0(\mathbf{W}_3 \otimes \mathbf{s}^\top) + \overline{\mathbf{V}}(\mathbf{I}_{n_3} \otimes \mathbf{s}^\top)]_1}_{\mathbf{C}_3}, \quad \text{kem} = [\overline{\mathbf{k}\mathbf{s}^\top}]_T \end{aligned}$$

– **KeyGen**(msk, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$): Sample

$$\mathbf{r}_2 \leftarrow \mathbb{Z}_p^k, \mathbf{r}_3 \leftarrow \mathbb{Z}_p^{(k+1)k'}$$

and output

$$\text{sk} = \left(\underbrace{[\mathbf{r}_2 \mathbf{B}]_2}_{\mathbf{d}_0}, \underbrace{[(\mathbf{x}_1 \otimes \mathbf{r}_2 \mathbf{B}) \mathbf{W}_1]_2}_{\mathbf{d}_1}, \underbrace{[\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{r}_2 \mathbf{B} \mathbf{W}_2]_2}_{\mathbf{d}_2}, \underbrace{[\mathbf{r}_3 + \mathbf{r}_2 \mathbf{B} \mathbf{V}_0]_2}_{\mathbf{d}_3}, \right. \\ \left. \underbrace{[\mathbf{x}_3 \otimes \mathbf{k} + \mathbf{r}_2 \mathbf{B} \mathbf{V}]_2}_{\mathbf{d}_4} \right)$$

– Dec(sk, $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, ct, f): Output

$$\left[(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \overbrace{(\mathbf{d}_3 \mathbf{C}_2 + \mathbf{d}_4 (\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) - \mathbf{d}_0 \mathbf{C}_3)}^{(i)}) \mathbf{f}^\top - (\mathbf{x}_1 \otimes \overbrace{(\mathbf{d}_2 (\mathbf{I}_{n_2} \otimes \mathbf{C}_2))}^{(ii)}) \mathbf{f}^\top \right. \\ \left. + \overbrace{(\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{c}_1^\top}^{(iii)} - \overbrace{\mathbf{d}_1 \mathbf{c}_0^\top}^{(iv)} \right]_{\mathcal{T}}^{((\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \mathbf{f}^\top)^{-1}}$$

where the terms in (i), (ii), (iii), (iv) are computed in \mathbb{G}_T using the pairing.

4.2 Correctness

Step 1. First, observe that we can rewrite ct, kem in terms of msk and $[\mathbf{c}_0]_1$ (where $\mathbf{c}_0^\top = \mathbf{A} \mathbf{s}^\top$), namely:

$$\text{ct} = \left(\underbrace{[\mathbf{A} \mathbf{s}^\top]_1}_{\mathbf{c}_0^\top}, \underbrace{[(\mathbf{I}_{n_1} \otimes \mathbf{W}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1})) (\mathbf{f}^\top \otimes \mathbf{I}_{k+1}) + \mathbf{W}_1] \mathbf{c}_0^\top}_1}_{\mathbf{c}_1^\top} \right) \quad (8) \\ \left[\underbrace{(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) (\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top)}_1, \underbrace{[(\mathbf{V}_0 (\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) + \mathbf{V}) (\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top)]}_1 \right], \\ \text{kem} = [\mathbf{k} \mathbf{c}_0^\top]_{\mathcal{T}}$$

To see that this is equivalent to the output of Enc, we will use

$$(\mathbf{I}_{k'} \otimes \mathbf{A})(\mathbf{W}_3 \otimes \mathbf{s}^\top) = (\mathbf{W}_3 \otimes \mathbf{I}_{k+1})(\mathbf{I}_{n_3} \otimes \mathbf{A} \mathbf{s}^\top) \quad (9)$$

We start with the first summand in \mathbf{c}_1^\top :

$$\begin{aligned} & \underbrace{(\mathbf{I}_{n_1} \otimes (\overbrace{\mathbf{W}_2 (\mathbf{I}_{n_2} \otimes \mathbf{I}_{k'} \otimes \mathbf{A})}^{\text{using (9)}}))}_{\mathbf{c}_1^\top} (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{s}^\top) \mathbf{f}^\top \\ &= (\mathbf{I}_{n_1} \otimes \mathbf{W}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1}) (\mathbf{I}_{n_2 n_3} \otimes \mathbf{A} \mathbf{s}^\top)) \mathbf{f}^\top \quad \text{using (9)} \\ &= (\mathbf{I}_{n_1} \otimes \mathbf{W}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1})) (\mathbf{I}_{n_1 n_2 n_3} \otimes \mathbf{A} \mathbf{s}^\top) (\mathbf{f}^\top \otimes \mathbf{1}) \\ &= (\mathbf{I}_{n_1} \otimes \mathbf{W}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1})) (\mathbf{f}^\top \otimes \mathbf{I}_{k+1}) \mathbf{A} \mathbf{s}^\top \end{aligned}$$

For the remaining terms, we have:

$$\begin{aligned}
\overline{\mathbf{W}}_1 \mathbf{s}^\top &= \mathbf{W}_1 \mathbf{A} \mathbf{s}^\top \\
\mathbf{W}_3 \otimes \mathbf{A} \mathbf{s}^\top &= (\mathbf{W}_3 \otimes \mathbf{I}_{k+1})(\mathbf{I}_{n_3} \otimes \mathbf{A} \mathbf{s}^\top) \\
&= \underbrace{\mathbf{V}_0(\mathbf{I}_{k'} \otimes \mathbf{A})}_{\overline{\mathbf{V}}_0} (\mathbf{W}_3 \otimes \mathbf{s}^\top) = (\mathbf{V}_0(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}))(\mathbf{I}_{n_3} \otimes \mathbf{A} \mathbf{s}^\top) \quad \text{using (9)} \\
\overline{\mathbf{V}}(\mathbf{I}_{n_3} \otimes \mathbf{s}^\top) &= \mathbf{V}(\mathbf{I}_{n_3} \otimes \mathbf{A} \mathbf{s}^\top)
\end{aligned}$$

Step 2. Next, we show that

$$\begin{aligned}
&(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \mathbf{f}^\top \cdot \mathbf{k} \mathbf{c}_0^\top \\
&= (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \underbrace{(\mathbf{d}_3 \mathbf{C}_2 + \mathbf{d}_4(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) - \mathbf{d}_0 \mathbf{C}_3)}_{(i)}) \mathbf{f}^\top - (\mathbf{x}_1 \otimes \underbrace{(\mathbf{d}_2(\mathbf{I}_{n_2} \otimes \mathbf{C}_2))}_{(ii)}) \mathbf{f}^\top + \underbrace{(\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{c}_1^\top}_{(iii)} - \underbrace{\mathbf{d}_1 \mathbf{c}_0^\top}_{(iv)}
\end{aligned}$$

This follows readily from the following calculations:

$$\begin{aligned}
(i) &= (\mathbf{r}_3 + \mathbf{d}_0 \mathbf{V}_0)(\mathbf{W}_3 \otimes \mathbf{I}_{k+1})(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) + (\mathbf{x}_3 \otimes \mathbf{k} + \mathbf{d}_0 \mathbf{V})(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) \\
&\quad - \mathbf{d}_0(\mathbf{V}_0(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) + \mathbf{V})(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) \\
&= (\mathbf{r}_3(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) + \mathbf{x}_3 \otimes \mathbf{k})(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) \\
&= \mathbf{r}_3(\mathbf{W}_3 \otimes \mathbf{c}_0^\top) + \mathbf{x}_3 \otimes \mathbf{k} \mathbf{c}_0^\top \\
(ii) &= (\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{d}_0 \mathbf{W}_2) \cdot (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{c}_0^\top) \\
&= \mathbf{x}_2 \otimes (\mathbf{r}_3(\mathbf{W}_3 \otimes \mathbf{c}_0^\top)) + \mathbf{d}_0 \mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{c}_0^\top) \\
(iii) &= (\mathbf{x}_1 \otimes \mathbf{d}_0)((\mathbf{I}_{n_1} \otimes \mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1}))(\mathbf{f}^\top \otimes \mathbf{I}_{k+1}) + \mathbf{W}_1) \mathbf{c}_0^\top \\
&= (\mathbf{x}_1 \otimes (\mathbf{d}_0 \mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{c}_0^\top))) \mathbf{f}^\top + (\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{W}_1 \mathbf{c}_0^\top \\
(iv) &= (\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{W}_1 \mathbf{c}_0^\top
\end{aligned}$$

and thus

$$\begin{aligned}
&(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \underbrace{(\mathbf{d}_3 \mathbf{C}_2 + \mathbf{d}_4(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top) - \mathbf{d}_0 \mathbf{C}_3)}_{(i)}) \mathbf{f}^\top - (\mathbf{x}_1 \otimes \underbrace{(\mathbf{d}_2(\mathbf{I}_{n_2} \otimes \mathbf{C}_2))}_{(ii)}) \mathbf{f}^\top \\
&+ \underbrace{(\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{c}_1^\top}_{(iii)} - \underbrace{\mathbf{d}_1 \mathbf{c}_0^\top}_{(iv)} \\
&= (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes (\mathbf{r}_3(\mathbf{W}_3 \otimes \mathbf{c}_0^\top))) \cdot \mathbf{f}^\top + (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3 \otimes \mathbf{k} \mathbf{c}_0^\top) \cdot \mathbf{f}^\top \\
&\quad - (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes (\mathbf{r}_3(\mathbf{W}_3 \otimes \mathbf{c}_0^\top))) \cdot \mathbf{f}^\top - (\mathbf{x}_1 \otimes \mathbf{d}_0 \mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{c}_0^\top)) \cdot \mathbf{f}^\top \\
&\quad + (\mathbf{x}_1 \otimes (\mathbf{d}_0 \mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{c}_0^\top))) \mathbf{f}^\top + (\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{W}_1 \mathbf{c}_0^\top \\
&\quad - (\mathbf{x}_1 \otimes \mathbf{d}_0) \mathbf{W}_1 \mathbf{c}_0^\top
\end{aligned}$$

Correctness then follows readily.

4.3 Core of Security Proof

As described in the technical overview in Sect. 2.3, the core of the security of lies in proving adaptive security of the scheme in (7) where the adversary is given just a single ciphertext and a single key and no mpk and with $s = r_2 = 1$. We formalize and prove this statement next.

Given $\alpha_0, \alpha_1 \in \mathbb{Z}_p$, we define the distribution \mathcal{D}_b over (ct, sk) where:

$$\begin{aligned} \text{ct} &= [(\mathbf{I}_{n_1} \otimes \mathbf{w}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3))\mathbf{f}^\top + \mathbf{w}_1^\top]_1, [\mathbf{W}_3]_1, [\mathbf{v}_0\mathbf{W}_3 + \mathbf{v}]_1, \\ \text{sk} &= [\mathbf{x}_1\mathbf{w}_1^\top]_2, [\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{w}_2]_2, [\mathbf{r}_3 + \mathbf{v}_0]_2, [\mathbf{x}_3\alpha_b + \mathbf{v}]_2 \end{aligned}$$

and

$$\mathbf{w}_1 \leftarrow \mathbb{Z}_p^{n_1}, \mathbf{w}_2 \leftarrow \mathbb{Z}_p^{k'n_2}, \mathbf{v} \leftarrow \mathbb{Z}_p^{n_3}, \mathbf{v}_0 \leftarrow \mathbb{Z}_p^{k'}, \mathbf{W}_3 \leftarrow \mathbb{Z}_p^{k' \times n_3}, \mathbf{r}_3 \leftarrow \mathbb{Z}_p^{k'}$$

and we allow adaptive choices of \mathbf{f} and $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ subject to the constraint $(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3)\mathbf{f}^\top = 0$.

Lemma 1. *For all $\alpha_0, \alpha_1 \in \mathbb{Z}_p$, we have $\mathcal{D}_0 \approx_c \mathcal{D}_1$, under the k' -Lin assumption in \mathbb{G}_1 and the bi- k' -Lin assumption.*

Proof. We bound the advantage of guessing b given $\mathcal{D}_b, b \leftarrow \{0, 1\}$ by a negligible function. We proceed via a case analysis, following the “doubly selective” framework [4, 20]:

Case 1 (selective). \mathbf{f} is queried before $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

Step 1. We start by sampling random $\tilde{\mathbf{w}}_1 \leftarrow \mathbb{Z}_p^{n_1}, \tilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{n_3}$ and programming

$$\tilde{\mathbf{w}}_1^\top = (\mathbf{I}_{n_1} \otimes \mathbf{w}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3))\mathbf{f}^\top + \mathbf{w}_1^\top, \quad \tilde{\mathbf{v}} = \mathbf{v}_0\mathbf{W}_3 + \mathbf{v}$$

We can then rewrite ct, sk as:

$$\begin{aligned} \text{ct} &= [\tilde{\mathbf{w}}_1^\top]_1, [\mathbf{W}_3]_1, [\tilde{\mathbf{v}}]_1 \\ \text{sk} &= [\mathbf{x}_1\tilde{\mathbf{w}}_1^\top - (\mathbf{x}_1 \otimes \mathbf{w}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3))\mathbf{f}^\top]_2, [\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{w}_2]_2, [\mathbf{r}_3 + \mathbf{v}_0]_2, \\ &\quad [\mathbf{x}_3\alpha_b + \tilde{\mathbf{v}} - \mathbf{v}_0\mathbf{W}_3]_2 \end{aligned}$$

Step 2. Next, we sample random $\tilde{\mathbf{w}}_2 \leftarrow \mathbb{Z}_p^{k'n_2}, \tilde{\mathbf{v}}_0 \leftarrow \mathbb{Z}_p^{k'}$ and program

$$\tilde{\mathbf{w}}_2 = \mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{w}_2, \quad \tilde{\mathbf{v}}_0 = \mathbf{r}_3 + \mathbf{v}_0$$

We can then rewrite ct, sk as:

$$\begin{aligned} \text{ct} &= [\tilde{\mathbf{w}}_1^\top]_1, [\mathbf{W}_3]_1, [\tilde{\mathbf{v}}]_1 \\ \text{sk} &= [\mathbf{x}_1\tilde{\mathbf{w}}_1^\top + (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{r}_3\mathbf{W}_3)\mathbf{f}^\top - (\mathbf{x}_1 \otimes \tilde{\mathbf{w}}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3))\mathbf{f}^\top]_2, [\tilde{\mathbf{w}}_2]_2, [\tilde{\mathbf{v}}_0]_2, \\ &\quad [\mathbf{x}_3\alpha_b + \mathbf{r}_3\mathbf{W}_3 + \tilde{\mathbf{v}} - \tilde{\mathbf{v}}_0\mathbf{W}_3]_2 \end{aligned}$$

Step 3. Next, by the bilateral k' -Lin assumption, we have:

$$\{[\mathbf{W}_3]_1, [\mathbf{r}_3\mathbf{W}_3 + \mathbf{x}_3\alpha_b]_2\} \approx_c \{[\mathbf{W}_3]_1, [\tilde{\mathbf{d}}_4]_2\}, \quad \tilde{\mathbf{d}}_4 \leftarrow \mathbb{Z}_p^{n_3},$$

This means that

$$\text{sk} \approx_c [\mathbf{x}_1 \tilde{\mathbf{w}}_1^\top + (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \tilde{\mathbf{d}}_4) \mathbf{f}^\top - \overbrace{(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \mathbf{f}^\top}^{=0} \alpha_b - (\mathbf{x}_1 \otimes \tilde{\mathbf{w}}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3)) \mathbf{f}^\top]_2, \\ [\tilde{\mathbf{w}}_2]_2, [\tilde{\mathbf{v}}_0]_2, [\tilde{\mathbf{d}}_4 + \tilde{\mathbf{v}} - \tilde{\mathbf{v}}_0 \mathbf{W}_3]_2$$

That is, the distribution \mathcal{D}_b is computationally indistinguishable from:

$$\text{ct} = [\tilde{\mathbf{w}}_1^\top]_1, [\mathbf{W}_3]_1, [\tilde{\mathbf{v}}]_1 \\ \text{sk} = [\mathbf{x}_1 \tilde{\mathbf{w}}_1^\top + (\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \tilde{\mathbf{d}}_4) \mathbf{f}^\top - (\mathbf{x}_1 \otimes \tilde{\mathbf{w}}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3)) \mathbf{f}^\top]_2, [\tilde{\mathbf{w}}_2]_2, [\tilde{\mathbf{v}}_0]_2, [\tilde{\mathbf{d}}_4 + \tilde{\mathbf{v}} - \tilde{\mathbf{v}}_0 \mathbf{W}_3]_2$$

which is independent of the bit b .

Case 2: (co-selective). $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_2$ is queried before \mathbf{f} .

Step 1. We start by sampling random $\tilde{\mathbf{v}}_0 \leftarrow \mathbb{Z}_p^{k'}$, $\tilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{n_3}$, $\tilde{\mathbf{w}}_2 \leftarrow \mathbb{Z}_p^{k' n_2}$ and programming

$$\tilde{\mathbf{w}}_2 = \mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{w}_2, \quad \tilde{\mathbf{v}}_0 = \mathbf{r}_3 + \mathbf{v}_0, \quad \tilde{\mathbf{v}} = \mathbf{x}_3 \alpha_b + \mathbf{v}$$

We can then rewrite ct, sk as:

$$\text{ct} = [-(\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \mathbf{r}_3 \mathbf{W}_3) \mathbf{f}^\top + \mathbf{w}_1^\top + (\mathbf{I}_{n_1} \otimes \tilde{\mathbf{w}}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3)) \mathbf{f}^\top]_1, \\ [\mathbf{W}_3]_1, [-(\mathbf{r}_3 \mathbf{W}_3 + \mathbf{x}_3 \alpha_b) + \tilde{\mathbf{v}}_0 \mathbf{W}_3 + \tilde{\mathbf{v}}]_1 \\ \text{sk} = [\mathbf{x}_1 \mathbf{w}_1^\top]_2, [\tilde{\mathbf{w}}_2]_2, [\tilde{\mathbf{v}}_0]_2, [\tilde{\mathbf{v}}]_2$$

Step 2. Next, by the k' -Lin assumption in \mathbb{G}_1 , we have:

$$\{[\mathbf{W}_3]_1, [\mathbf{r}_3 \mathbf{W}_3 + \mathbf{x}_3 \alpha_b]_1\} \approx_c \{[\mathbf{W}_3]_1, [\tilde{\mathbf{c}}_3]_1\}, \quad \tilde{\mathbf{c}}_3 \leftarrow \mathbb{Z}_p^{n_3},$$

This means that

$$\text{ct} \approx_c [(\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \mathbf{x}_3 \alpha_b) \mathbf{f}^\top + \mathbf{w}_1^\top - (\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \tilde{\mathbf{c}}_3) \mathbf{f}^\top \\ + (\mathbf{I}_{n_1} \otimes \tilde{\mathbf{w}}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3)) \mathbf{f}^\top]_1, [\mathbf{W}_3]_1, [-\tilde{\mathbf{c}}_3 + \tilde{\mathbf{v}}_0 \mathbf{W}_3 + \tilde{\mathbf{v}}]_1$$

Step 3. At this point, the view of the adversary is given by:

$$\text{ct} = \left[(\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \mathbf{x}_3 \alpha_b) \mathbf{f}^\top + \mathbf{w}_1^\top \right] - (\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \tilde{\mathbf{c}}_3) \mathbf{f}^\top \\ + (\mathbf{I}_{n_1} \otimes \tilde{\mathbf{w}}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3)) \mathbf{f}^\top]_1, [\mathbf{W}_3]_1, [-\tilde{\mathbf{c}}_3 + \tilde{\mathbf{v}}_0 \mathbf{W}_3 + \tilde{\mathbf{v}}]_1 \\ \text{sk} = \left[\mathbf{x}_1 \mathbf{w}_1^\top \right]_2, [\tilde{\mathbf{w}}_2]_2, [\tilde{\mathbf{v}}_0]_2, [\tilde{\mathbf{v}}]_2$$

where all of the leakage on α_b comes from the boxed terms. We claim that the advantage of the adversary is 0 here. It suffices to prove this for the case \mathbf{f} is fixed in advance; then, a random guessing (also referred to as complexity leveraging) argument tells us that the advantage is still 0 even for an adaptively chosen \mathbf{f} .

Sample a random $\tilde{\mathbf{w}}_1 \leftarrow \mathbb{Z}_p^{n_1}$ and program

$$\tilde{\mathbf{w}}_1 = (\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \mathbf{x}_3 \alpha_b) \mathbf{f}^\top + \mathbf{w}_1^\top$$

Then, we can write

$$\mathbf{x}_1 \mathbf{w}_1^\top = \mathbf{x}_1 \tilde{\mathbf{w}}_1^\top - \mathbf{x}_1 (\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \mathbf{x}_3 \alpha_b) \mathbf{f}^\top = \mathbf{x}_1 \tilde{\mathbf{w}}_1^\top - \overbrace{(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \mathbf{f}^\top}^{=0} \alpha_b$$

This means that the view of the adversary (for a fixed \mathbf{f}) is identically distributed to

$$\begin{aligned} \text{ct} &= \left[\tilde{\mathbf{w}}_1^\top \right] - (\mathbf{I}_{n_1} \otimes \mathbf{x}_2 \otimes \tilde{\mathbf{c}}_3) \mathbf{f}^\top + (\mathbf{I}_{n_1} \otimes \tilde{\mathbf{w}}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3)) \mathbf{f}^\top \Big|_1, [\mathbf{W}_3]_1, [-\tilde{\mathbf{c}}_3 + \tilde{\mathbf{v}}_0 \mathbf{W}_3 + \tilde{\mathbf{v}}]_1 \\ \text{sk} &= \left[\mathbf{x}_1 \tilde{\mathbf{w}}_1^\top \right]_2, [\tilde{\mathbf{w}}_2]_2, \qquad \qquad \qquad [\tilde{\mathbf{v}}_0]_2, [\tilde{\mathbf{v}}]_2 \end{aligned}$$

The above distribution is independent of the bit b , and hence the advantage is 0.

4.4 Security Proof

The rest of the proof is a routine application of the dual system encryption methodology [4, 10, 19, 20, 24, 25], apart from the substitutions in (11), which slightly generalizes that in [10], as described at the end of Sect. 2.4.

Auxiliary distributions. We define the following additional ciphertext and key distributions used in the security proof. Sample $\delta \leftarrow \mathbb{Z}_p$.

- $(\hat{\text{ct}}, \hat{\text{kem}})$ is the same as (ct, kem) in (8), except we replace $\mathbf{A} \mathbf{s}^\top$ with $\mathbf{c}^\top \leftarrow \mathbb{Z}_p^{(k+1) \times 1}$:

$$\begin{aligned} \hat{\text{ct}} &= \left([\mathbf{c}^\top]_1, [((\mathbf{I}_{n_1} \otimes \mathbf{W}_2 (\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1})) (\mathbf{f}^\top \otimes \mathbf{I}_{k+1}) + \mathbf{W}_1) \mathbf{c}^\top]_1, \right. \\ &\quad \left. [(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) (\mathbf{I}_{n_3} \otimes \mathbf{c}^\top)]_1, [(\mathbf{V}_0 (\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) + \mathbf{V}) (\mathbf{I}_{n_3} \otimes \mathbf{c}^\top)]_1 \right) \\ \hat{\text{kem}} &= [\mathbf{k} \mathbf{c}^\top]_T \end{aligned}$$

Henceforth, let $\mathbf{a}^\perp \in \mathbb{Z}_p^{k+1}$ satisfying $\mathbf{a}^\perp \cdot \mathbf{A} = \mathbf{0}$, $\mathbf{a}^\perp \cdot \mathbf{c}^\top = 1$, which exists with probability $1 - 1/p$ over \mathbf{c} .

- $\hat{\text{sk}}$ is the same as sk except we replace \mathbf{k} with $\mathbf{k} + \delta \mathbf{a}^\perp$:

$$\hat{\text{sk}} = \left(\underbrace{[\mathbf{r}_2 \mathbf{B}]_2}_{\mathbf{d}_0}, \underbrace{[(\mathbf{x}_1 \otimes \mathbf{r}_2 \mathbf{B}) \mathbf{W}_1]_2}_{\mathbf{d}_1}, \underbrace{[\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{r}_2 \mathbf{B} \mathbf{W}_2]_2}_{\mathbf{d}_2}, \underbrace{[\mathbf{r}_3 + \mathbf{r}_2 \mathbf{B} \mathbf{V}_0]_2}_{\mathbf{d}_3}, \underbrace{[\mathbf{x}_3 \otimes (\mathbf{k} + \delta \mathbf{a}^\perp) + \mathbf{r}_2 \mathbf{B} \mathbf{V}]_2}_{\mathbf{d}_4} \right)$$

- $\text{sk}[1]$ is the same as sk except we replace $\mathbf{r}_2 \mathbf{B}$ with $\mathbf{d} \leftarrow \mathbb{Z}_p^{k+1}$:

$$\text{sk}[1] = \left(\underbrace{[\mathbf{d}]_2}_{\mathbf{d}_0}, \underbrace{[(\mathbf{x}_1 \otimes \mathbf{d}) \mathbf{W}_1]_2}_{\mathbf{d}_1}, \underbrace{[\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{d} \mathbf{W}_2]_2}_{\mathbf{d}_2}, \underbrace{[\mathbf{r}_3 + \mathbf{d} \mathbf{V}_0]_2}_{\mathbf{d}_3}, \underbrace{[\mathbf{x}_3 \otimes \mathbf{k} + \mathbf{d} \mathbf{V}]_2}_{\mathbf{d}_4} \right)$$

- $\text{sk}[2]$ is the same as $\text{sk}[1]$ except we replace \mathbf{k} with $\mathbf{k} + \delta \mathbf{a}^\perp$:

$$\text{sk}[2] = \left(\underbrace{[\mathbf{d}]_2}_{\mathbf{d}_0}, \underbrace{[(\mathbf{x}_1 \otimes \mathbf{d}) \mathbf{W}_1]_2}_{\mathbf{d}_1}, \underbrace{[\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{d} \mathbf{W}_2]_2}_{\mathbf{d}_2}, \underbrace{[\mathbf{r}_3 + \mathbf{d} \mathbf{V}_0]_2}_{\mathbf{d}_3}, \underbrace{[\mathbf{x}_3 \otimes (\mathbf{k} + \delta \mathbf{a}^\perp) + \mathbf{d} \mathbf{V}]_2}_{\mathbf{d}_4} \right)$$

Following the terminology in prior works, $(\hat{\text{ct}}, \hat{\text{kem}})$ is the semi-functional (SF) ciphertext; $\hat{\text{sk}}$ is the SF secret key; $\text{sk}[1]$ is the pseudo-normal secret key, and $\text{sk}[2]$ is the pseudo-SF secret key.

Game sequence. We present a series of games. We write Adv_{xx} to denote the advantage of \mathcal{A} in Game_{xx} . Suppose \mathcal{A} makes q queries to KeyGen : let $(\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i)$ denote the i 'th query, and let one of $\text{sk}^i, \text{sk}^i[1], \text{sk}^i[2], \hat{\text{sk}}^i$ denote the i 'th key.

- Game_0 : is the real security game.
- Game_1 : is the same as Game_0 except we replace (ct, kem) with $(\hat{\text{ct}}, \hat{\text{kem}})$.
- $\text{Game}_{2,i}$ for $i = 1, \dots, q$: is the same as Game_1 , except the first $i - 1$ keys are given by $\hat{\text{sk}}^1, \dots, \hat{\text{sk}}^{i-1}$ (semi-functional) and the last $q - i$ keys are given by $\text{sk}^{i+1}, \dots, \text{sk}^q$ (normal). There are 4 sub-games, where the i 'th key transitions from sk^i in $\text{Game}_{2,i,0}$, to $\text{sk}^i[1]$ in $\text{Game}_{2,i,1}$, to $\text{sk}^i[2]$ in $\text{Game}_{2,i,2}$, to $\hat{\text{sk}}^i$ in $\text{Game}_{2,i,3}$. Note that $\text{Game}_1 = \text{Game}_{2,1,0}$ and $\text{Game}_{2,i,3} = \text{Game}_{2,(i+1),0}$.
- Game_3 : is the same as $\text{Game}_{2,q,3}$, except that $\text{kem}_0 \leftarrow_{\mathbb{R}} \mathbb{G}_T$.

In Game_3 , the view of \mathcal{A} is statistically independent of the challenge bit b . Hence, $\text{Adv}_3 = 0$. We complete the proof by establishing the following claims:

$\text{Game}_0 \approx_c \text{Game}_1$. This follows readily from the k -Lin assumption in \mathbb{G}_1 , where the reduction on input $[\mathbf{A}]_1, [\mathbf{c}_0]_1$ where $\mathbf{c}_0^\top \in \{\mathbf{A}\mathbf{s}^\top, \mathbf{c}^\top\}$, $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}$:

- runs the honest Setup to generate all the terms in (mpk, msk) apart from \mathbf{A} ;
- uses $\text{msk}, \mathbf{c}_0^\top$ to compute the challenge ciphertext and KEM:

$$\begin{aligned} & ([\mathbf{c}_0^\top]_1, [((\mathbf{I}_{n_1} \otimes \mathbf{W}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1}))(\mathbf{f}^\top \otimes \mathbf{I}_{k+1}) + \mathbf{W}_1)\mathbf{c}_0^\top]_1, \\ & [(\mathbf{W}_3 \otimes \mathbf{I}_{k+1})(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top)]_1, [(\mathbf{V}_0(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) + \mathbf{V})(\mathbf{I}_{n_3} \otimes \mathbf{c}_0^\top)]_1) \\ & [\mathbf{kc}_0^\top]_T \end{aligned}$$

By (8), this is (ct, kem) when $\mathbf{c}_0^\top = \mathbf{A}\mathbf{s}^\top$, and $(\hat{\text{ct}}, \hat{\text{kem}})$ when $\mathbf{c}_0^\top = \mathbf{c}^\top$;

- uses msk to simulate the KeyGen oracle.

$\text{Game}_{2,i,0} \approx_c \text{Game}_{2,i,1}$, $\text{Game}_{2,i,2} \approx_c \text{Game}_{2,i,3}$. This follows readily from the k -Lin assumption in \mathbb{G}_2 , where the reduction on input $[\mathbf{B}]_1, [\mathbf{d}_0]_1$ where $\mathbf{d}_0 \in \{\mathbf{rB}, \mathbf{d}\}$, $\mathbf{d} \leftarrow \mathbb{Z}_p^{k+1}$:

- runs the honest Setup to generate all the terms in (mpk, msk) apart from \mathbf{B} ;
- samples a random $\delta \in \mathbb{Z}_p$;
- samples a random $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}$ and uses msk, \mathbf{c} to compute the challenge ciphertext using (8);
- uses msk and δ to generate the first $i - 1$ keys $\hat{\text{sk}}^1, \dots, \hat{\text{sk}}^{i-1}$ and the last $q - i$ keys $\text{sk}^{i+1}, \dots, \text{sk}^q$;
- computes the i 'th key using $[\mathbf{d}_0]$ and msk, δ using:

$$([\mathbf{d}_0]_2, [(\mathbf{x}_1^i \otimes \mathbf{d}_0)\mathbf{W}_1]_2, [\mathbf{x}_2^i \otimes \mathbf{r}_3 + \mathbf{d}_0\mathbf{W}_2]_2, [\mathbf{r}_3 + \mathbf{d}_0\mathbf{V}_0]_2, [\mathbf{x}_3^i \otimes \mathbf{k} + \mathbf{d}_0\mathbf{V}]_2)$$

This is sk^i when $\mathbf{d}_0 = \mathbf{rB}$, and $\text{sk}^i[1]$ when $\mathbf{d}_0 = \mathbf{d}$.

Game_{2.i.1} \approx_c **Game**_{2.i.2}. To prove **Game**_{2.i.1} \approx_c **Game**_{2.i.2}, it suffices to show

$$(\text{aux}, \hat{\text{ct}}, \text{sk}^i[1]) \approx_c (\text{aux}, \hat{\text{ct}}, \text{sk}^i[2]) \quad (10)$$

where

$$\text{aux} := (\mathbb{G}, \mathbf{A}, \mathbf{c}, \mathbf{B}, \mathbf{k}, \delta, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_1, \overline{\mathbf{V}}, \overline{\mathbf{V}}_0, \mathbf{B}\mathbf{W}_2, (\mathbf{I}_{n_1} \otimes \mathbf{B})\mathbf{W}_1, \mathbf{B}\mathbf{V}, \mathbf{B}\mathbf{V}_0)$$

This is because given aux , we can compute mpk , $\hat{\text{kem}}$ as well as both sk (for the last $q - i$ key queries) and $\hat{\text{sk}}$ (for the first $i - 1$ key queries).

– To compute sk , we sample $\mathbf{r}_2 \leftarrow \mathbb{Z}_p^k$, $\mathbf{r}_3 \leftarrow \mathbb{Z}_p^{(k+1)k'}$ and output

$$\text{sk} = (\underbrace{[\mathbf{r}_2 \mathbf{B}]_2}_{\mathbf{d}_0}, \underbrace{[(\mathbf{x}_1 \otimes \mathbf{r}_2) \cdot (\mathbf{I}_{n_1} \otimes \mathbf{B})\mathbf{W}_1]_2}_{\mathbf{d}_1}, \underbrace{[\mathbf{x}_2 \otimes \mathbf{r}_3 + \mathbf{r}_2 \cdot \mathbf{B}\mathbf{W}_2]_2}_{\mathbf{d}_2}, \underbrace{[\mathbf{r}_3 + \mathbf{r}_2 \cdot \mathbf{B}\mathbf{V}_0]_2}_{\mathbf{d}_3}, \underbrace{[\mathbf{x}_3 \otimes \mathbf{k} + \mathbf{r}_2 \cdot \mathbf{B}\mathbf{V}]_2}_{\mathbf{d}_4})$$

– To compute $\hat{\text{sk}}$, we would first compute \mathbf{a}^\perp given \mathbf{A}, \mathbf{c} , and then proceed as in sk , except we replace \mathbf{k} with $\mathbf{k} + \delta \mathbf{a}^\perp$.

We proceed to prove (10) using Lemma 1. Henceforth, let $\mathbf{b}^\perp \in \mathbb{Z}_p^{k+1}$ satisfying $\mathbf{B} \cdot \mathbf{b}^{\perp\top} = \mathbf{0}$, $\mathbf{d} \cdot \mathbf{b}^{\perp\top} = 1$, which exists with probability $1 - 1/p$ over \mathbf{d} , where $[\mathbf{d}]_2$ is the first component of $\text{sk}^i[1]$ and $\text{sk}^i[2]$. Sample

$$\begin{aligned} \mathbf{W}'_1 &\leftarrow \mathbb{Z}_p^{(k+1)n_1 \times (k+1)}, & \mathbf{W}'_2 &\leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)k'n_2}, & \mathbf{V}' &\leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)n_3}, \\ \mathbf{w}_1 &\leftarrow \mathbb{Z}_p^{n_1}, & \mathbf{w}_2 &\leftarrow \mathbb{Z}_p^{k'n_2}, & \mathbf{v} &\leftarrow \mathbb{Z}_p^{n_3}, \\ \mathbf{k}' &\leftarrow \mathbb{Z}_p^{k+1} & \mathbf{r}'_3 &\leftarrow \mathbb{Z}_p^{(k+1)k'} \\ \alpha &\leftarrow \mathbb{Z}_p, & \mathbf{r}_3 &\leftarrow \mathbb{Z}_p^{k'} \\ \mathbf{V}'_0 &\leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)k'} \\ \mathbf{v}_0 &\leftarrow \mathbb{Z}_p^{k'} \end{aligned}$$

and substitute

$$\begin{aligned} \mathbf{W}_1 &\mapsto \mathbf{W}'_1 + (\mathbf{I}_{n_1} \otimes \mathbf{b}^{\perp\top}) \cdot \mathbf{w}_1^\top \cdot \mathbf{a}^\perp \\ \mathbf{W}_2 &\mapsto \mathbf{W}'_2 + \mathbf{b}^{\perp\top} \cdot \mathbf{w}_2 \cdot (\mathbf{I}_{n_2} \otimes \mathbf{a}^\perp) \\ \mathbf{V} &\mapsto \mathbf{V}' + \mathbf{b}^{\perp\top} \cdot \mathbf{v} \cdot (\mathbf{I}_{n_3} \otimes \mathbf{a}^\perp) \\ \mathbf{V}_0 &\mapsto \mathbf{V}'_0 + \mathbf{b}^{\perp\top} \cdot \mathbf{v}_0 \cdot (\mathbf{I}_{k'} \otimes \mathbf{a}^\perp) \\ \mathbf{k} &\mapsto \mathbf{k}' + \alpha \cdot \mathbf{a}^\perp \\ \mathbf{r}_3 &\mapsto \mathbf{r}'_3 + \mathbf{r}_3 (\mathbf{I}_{k'} \otimes \mathbf{a}^\perp) \end{aligned} \quad (11)$$

where in the last line, we have $\mathbf{r}_3 \in \mathbb{Z}_p^{(k+1)k'}$ on the left, and $\mathbf{r}_3 \in \mathbb{Z}_p^{k'}$ on the right. We can then write

$$\text{aux} = (\mathbb{G}, \mathbf{A}, \mathbf{c}, \mathbf{B}, \mathbf{k}, \overline{\mathbf{W}}'_2, \overline{\mathbf{W}}'_1, \overline{\mathbf{V}}', \overline{\mathbf{V}}'_0, \mathbf{B}\mathbf{W}'_2, (\mathbf{I}_{n_1} \otimes \mathbf{B})\mathbf{W}'_1, \mathbf{B}\mathbf{V}', \mathbf{B}\mathbf{V}'_0)$$

and

$$\begin{aligned}
\hat{\mathbf{ct}} &= [\mathbf{c}^\top]_1, [((\mathbf{I}_{n_1} \otimes \mathbf{W}'_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3 \otimes \mathbf{I}_{k+1}))(\mathbf{f}^\top \otimes \mathbf{I}_{k+1}) + \mathbf{W}'_1)\mathbf{c}^\top \\
&\quad + (\mathbf{I}_{n_1} \otimes \mathbf{b}^{\perp\top}) \cdot \boxed{(\mathbf{I}_{n_1} \otimes \mathbf{w}_2(\mathbf{I}_{n_2} \otimes \mathbf{W}_3))\mathbf{f}^\top + \mathbf{w}_1^\top}]_1, \\
&\quad [(\boxed{\mathbf{W}_3} \otimes \mathbf{I}_{k+1})(\mathbf{I}_{n_3} \otimes \mathbf{c}^\top)]_1, [(\mathbf{V}'_0(\mathbf{W}_3 \otimes \mathbf{I}_{k+1}) + \mathbf{V}')(\mathbf{I}_{n_3} \otimes \mathbf{c}^\top) + \mathbf{b}^{\perp\top} \cdot \boxed{\mathbf{v}_0\mathbf{W}_3 + \mathbf{v}}]_1 \\
\hat{\mathbf{kem}} &= [\mathbf{k}'\mathbf{c}^\top + \alpha]_T \\
\mathbf{sk}^i[1] &= [\mathbf{d}]_2, [(\mathbf{x}_1^i \otimes \mathbf{d})\mathbf{W}'_1 + \boxed{\mathbf{x}_1^i\mathbf{w}_1^\top} \cdot \mathbf{a}^\perp]_2 \\
&\quad [(\mathbf{x}_2^i \otimes \mathbf{r}'_3 + \mathbf{d}\mathbf{W}'_2 + \boxed{\mathbf{x}_2^i \otimes \mathbf{r}_3 + \mathbf{w}_2}) \cdot (\mathbf{I}_{n_2} \otimes \mathbf{a}^\perp)]_2, \\
&\quad [\mathbf{r}'_3 + \mathbf{d}\mathbf{V}'_0 + \boxed{\mathbf{r}_3 + \mathbf{v}_0}] \cdot (\mathbf{I}_{k'} \otimes \mathbf{a}^\perp)]_2, \\
&\quad [\mathbf{x}_3^i \otimes \mathbf{k}' + \mathbf{d}\mathbf{V}' + \boxed{\mathbf{x}_3^i\alpha + \mathbf{v}}] \cdot (\mathbf{I}_{n_3} \otimes \mathbf{a}^\perp)]_2 \\
\mathbf{sk}^i[2] &= [\mathbf{d}]_2, [(\mathbf{x}_1^i \otimes \mathbf{d})\mathbf{W}'_1 + \boxed{\mathbf{x}_1^i\mathbf{w}_1^\top} \cdot \mathbf{a}^\perp]_2 \\
&\quad [(\mathbf{x}_2^i \otimes \mathbf{r}'_3 + \mathbf{d}\mathbf{W}'_2 + \boxed{\mathbf{x}_2^i \otimes \mathbf{r}_3 + \mathbf{w}_2}) \cdot (\mathbf{I}_{n_2} \otimes \mathbf{a}^\perp)]_2, \\
&\quad [\mathbf{r}'_3 + \mathbf{d}\mathbf{V}'_0 + \boxed{\mathbf{r}_3 + \mathbf{v}_0}] \cdot (\mathbf{I}_{k'} \otimes \mathbf{a}^\perp)]_2, \\
&\quad [\mathbf{x}_3^i \otimes \mathbf{k}' + \mathbf{d}\mathbf{V}' + \boxed{\mathbf{x}_3^i(\alpha + \delta) + \mathbf{v}}] \cdot (\mathbf{I}_{n_3} \otimes \mathbf{a}^\perp)]_2
\end{aligned}$$

Given the boxed terms together with $(\mathbf{c}, \mathbf{d}, \mathbf{W}'_1, \mathbf{W}'_2, \mathbf{V}', \mathbf{V}'_0, \mathbf{k}', \alpha, \mathbf{a}^\perp, \mathbf{b}^\perp, \delta, \mathbf{r}'_3)$, we can simulate $\hat{\mathbf{ct}}, \mathbf{sk}^i[1], \mathbf{sk}^i[2]$ as well as \mathbf{aux} . Therefore, it suffices to show that the boxed terms in $\mathbf{Game}_{2.i.1}$ and $\mathbf{Game}_{2.i.2}$ are computationally indistinguishable, which follows from Lemma 1. Concretely, the reduction on input $(\mathbf{ct}, \mathbf{sk})$ from \mathcal{D}_b corresponding to \mathbf{f} and $(\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i)$ and where $\alpha_0 = \alpha, \alpha_1 = \alpha + \delta$:

1. samples random $\mathbf{A}, \mathbf{B}, \mathbf{c}, \mathbf{d}, \mathbf{W}'_1, \mathbf{W}'_2, \mathbf{V}', \mathbf{V}'_0, \mathbf{k}', \alpha, \delta, \mathbf{r}'_3$, and call these values \mathbf{aux}' ;
2. computes $\mathbf{a}^\perp, \mathbf{b}^\perp$ using $\mathbf{A}, \mathbf{c}, \mathbf{B}, \mathbf{d}$;
3. computes \mathbf{aux} using $\mathbf{aux}', \mathbf{a}^\perp, \mathbf{b}^\perp$, which it then uses to compute \mathbf{mpk} as well as the first $i-1$ and the last $q-i$ key queries;
4. computes $\hat{\mathbf{ct}}$ by using \mathbf{ct} from \mathcal{D}_b for the boxed terms, and computing the remaining non-boxed terms using $\mathbf{aux}', \mathbf{a}^\perp, \mathbf{b}^\perp$;
5. computes $\hat{\mathbf{kem}}$ using \mathbf{aux}' ;
6. computes either $\mathbf{sk}^i[1]$ or $\mathbf{sk}^i[2]$ by using \mathbf{sk} from \mathcal{D}_b for the boxed terms, and computing the remaining non-boxed terms using $\mathbf{aux}', \mathbf{a}^\perp, \mathbf{b}^\perp$;

The output of the reduction is exactly $\mathbf{Game}_{2.i.(b+1)}$.

$\mathbf{Game}_{2.i.2} \approx_c \mathbf{Game}_{2.i.3}$. Analogous to $\mathbf{Game}_{2.i.0} \approx_c \mathbf{Game}_{2.i.1}$.

$\mathbf{Game}_{2.q.3} \equiv \mathbf{Game}_3$. In $\mathbf{Game}_{2,q}$, we have $\mathbf{kem}_0 = [\mathbf{k}\mathbf{c}^\top]$, whereas \mathbf{mpk} only leaks $[\mathbf{k}\mathbf{A}]_T$ and $\hat{\mathbf{sk}}^1, \dots, \hat{\mathbf{sk}}^q$ only leaks $\mathbf{k} + \delta\mathbf{a}^\perp$. The claim follows from the fact that $\mathbf{k}\mathbf{c}^\top$ is uniformly random in \mathbb{Z}_p given $\mathbf{k}\mathbf{A}$ and $\mathbf{k} + \delta\mathbf{a}^\perp$.

5 Broadcast Encryption with Size $N^{1/3}$

We can encode broadcast encryption for N parties as CP-ABE for degree 3 polynomials whenever $n_1 n_2 n_3 \geq N$, by using the folklore encoding of set membership in $S \subseteq [N]$ as a degree 3 polynomial over $\{0, 1\}^{n_1} \times \{0, 1\}^{n_2} \times \{0, 1\}^{n_3}$:

- given a set $S \subseteq [N]$, let $\mathbf{f} = (f_1, \dots, f_N) \in \{0, 1\}^N$ denote the characteristic vector for the set S (that is, $f_i = 1$ iff $i \in S$);
- given $y \in [N]$, we can pick $\mathbf{x}_1 \in \{0, 1\}^{n_1}, \mathbf{x}_2 \in \{0, 1\}^{n_2}, \mathbf{x}_3 \in \{0, 1\}^{n_3}$ such that $\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3 \in \{0, 1\}^{n_1 n_2 n_3}$ is the characteristic vector of the set $\{y\}$.
- then, $(\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3) \mathbf{f}^T = 1$ iff $y \in S$.

We can then set $n_1 = N^\delta, n_2 = N^{1-2\delta}, n_3 = N^\delta$ for any $0 \leq \delta \leq 1/3$, which yields

$$|\text{mpk}| = O(N^{1-2\delta}), |\text{ct}| = O(N^\delta), |\text{sk}| = O(N^{1-2\delta})$$

In particular, when $\delta = 1/3$, we achieve

$$|\text{mpk}| = O(N^{1/3}), |\text{ct}| = O(N^{1/3}), |\text{sk}| = O(N^{1/3})$$

A concrete example. While the main focus of this work is on asymptotically more efficient pairing-based broadcast encryption, our scheme does achieve pretty concrete good efficiency. We can instantiate our scheme with the popular BLS12-381 curve with $|\mathbb{G}_1|$ being 48 bytes and $|\mathbb{G}_2|$ being 96 bytes. Now, recall an application for broadcast encryption in BGW05 [6], namely file sharing in encrypted file systems. The Windows EFS has a limit of 256KB in the file header for the EFS meta-data, and supports a maximum of 800 individual users. Assuming 32-bit users IDs, we can support 1000 users with a file header (S, ct) of size $4 \times 1000 + 82 \times 48 = 7936$ bytes, where each user holds a secret key of size $67 \times 96 = 6432$ bytes. We can do slightly better by setting $n_1 = 20, n_2 = 10, n_3 = 5$, which yields a header of size $4 \times 1000 + 72 \times 48 = 7456$ bytes and a secret key of size $57 \times 96 = 5482$ bytes. However, since $N = 1000$ is fairly small, the broadcast encryption scheme with $O(\sqrt{N})$ parameters would also achieve similar performances: a file header of size $4 \times 1000 + 66 \times 48 = 7168$ and a secret key of size $68 \times 96 = 6528$ bytes.

Acknowledgments. I am extremely grateful to Junqing Gong for meticulous proof-reading and constructive feedback. I would also like to thank Tianren Liu for helpful discussions on the challenges of extending our $N^{1/3}$ CDS scheme in [22] to general fields while preserving degree 2 reconstruction.




References

1. Abdalla, M., Gong, J., Wee, H.: Functional Encryption for Attribute-Weighted Sums from k -Lin. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 685–716. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_23
2. Agrawal, S., Wichs, D., Yamada, S.: Optimal broadcast encryption from LWE and pairings in the standard model. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 149–178. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_6
3. Agrawal, S., Yamada, S.: Optimal broadcast encryption from pairings and LWE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. Part I, volume 12105 of LNCS, pp. 13–43. Springer, Heidelberg (May 2020)
4. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (May 2014)
5. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society Press, May 2007
6. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (Aug. 2005)
7. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: Juels, A., Wright, R.N., De Capitani, S., di Vimercati (eds.) ACM CCS 2006, pp. 211–220. ACM Press, October/November 2006
8. Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. Part I, volume 8616 of LNCS, pp. 206–223. Springer, Heidelberg (2014)
9. Brakerski, Z., Vaikuntanathan, V.: Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. Cryptology ePrint Archive, Report 2020/191 (2020)
10. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. Part II, volume 9057 of LNCS, pp. 595–624. Springer, Heidelberg (Apr. 2015)
11. 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. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8
12. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (Aug. 1994)
13. Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. Part I, volume 12110 of LNCS, pp. 95–120. Springer, Heidelberg (May 2020)
14. Gay, R., Kerenidis, I., Wee, H.: Communication complexity of conditional disclosure of secrets and attribute-based encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 485–502. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_24
15. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (Apr. 2009)

16. Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) 50th ACM STOC, pp. 660–670. ACM Press, June 2018
17. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006, pp. 89–98. ACM Press, October/November 2006. Available as Cryptology ePrint Archive Report 2006/309
18. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (Apr. 2008)
19. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_27
20. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_12
21. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. Part I, volume 10401 of LNCS, pp. 599–629. Springer, Heidelberg (Aug. 2017)
22. Liu, T., Vaikuntanathan, V., Wee, H.: Conditional disclosure of secrets via non-linear reconstruction. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 758–790. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_25
23. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (May 2005)
24. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (Aug. 2009)
25. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (Feb. 2014)
26. Wee, H.: Functional encryption for quadratic functions from k -Lin, revisited. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 210–228. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_8
27. Zhandry, M.: New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2020. Part I, LNCS, pp. 652–682. Springer, Heidelberg (Aug. 2020)



Fine-Grained Secure Attribute-Based Encryption

Yuyu Wang¹ , Jiaxin Pan² , and Yu Chen^{3,4,5} 

¹ University of Electronic Science and Technology of China, Chengdu, China
wangyuyu@uestc.edu.cn

² Department of Mathematical Sciences, NTNU - Norwegian University of Science and Technology, Trondheim, Norway
jiaxin.pan@ntnu.no

³ School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
yuchen@sdu.edu.cn

⁴ State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

⁵ Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao 266237, China

Abstract. Fine-grained cryptography is constructing cryptosystems in a setting where an adversary's resource is a-priori bounded and an honest party has less resource than an adversary. Currently, only simple form of encryption schemes, such as secret-key and public-key encryption, are constructed in this setting.

In this paper, we enrich the available tools in fine-grained cryptography by proposing the *first* fine-grained secure attribute-based encryption (ABE) scheme. Our construction is adaptively secure under the widely accepted worst-case assumption, $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$, and it is presented in a generic manner using the notion of predicate encodings (Wee, TCC'14). By properly instantiating the underlying encoding, we can obtain different types of ABE schemes, including identity-based encryption. Previously, all of these schemes were unknown in fine-grained cryptography. Our main technical contribution is constructing ABE schemes without using pairing or the Diffie-Hellman assumption. Hence, our results show that, even if one-way functions do not exist, we still have ABE schemes with meaningful security. For more application of our techniques, we construct an efficient (quasi-adaptive) non-interactive zero-knowledge (QA-NIZK) proof system.

Keywords: Fine-grained cryptography · Identity-based encryption · Attribute-based encryption · Quasi-adaptive non-interactive zero-knowledge proof

1 Introduction

1.1 Motivation

Modern cryptography bases the security of schemes on assumptions, including the basic ones (such as the existence of one-way functions (OWFs)), the more advanced ones (such as the hardness of factoring, discrete logarithms, and some lattice problems), and the much more exotic ones (such as the existence of generic groups [25, 29] or algebraic groups [17]). Although there is some analysis on these assumptions, it is less desirable. We are interested in how to construct cryptography based on much mild assumptions or which form of security cryptography can be achieved if all classical assumptions (such as the existence of OWFs) do not hold.

Fine-grained cryptography is a direction in approaching the aforementioned problems. It aims at cryptography with weaker security in a setting where adversaries have only bounded resources and honest users have less resources than the adversaries. Under this setting it is possible to make the underlying assumption extremely mild, for instance, assuming $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$. This is a widely accepted worst-case assumption. As $\oplus\text{L}/\text{poly}$ is the class of languages with polynomial-sized branching programs and all languages in NC^1 have polynomial-sized branching programs of constant width [3], this assumption holds if there exists one language having only polynomial-sized branching programs of non-constant width. This is different to assuming the existence of OWFs which is an average-case assumption. It requires that the OWF be hard to invert on a random input. Hence, $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$ is more likely to be true.

The study on fine-grained cryptography was initialized by Merkle [26]. In the recent years, we are interested in which kind of cryptosystems can be constructed in this setting. We highlight the recent constructions of OWFs [8], symmetric-key and (additively homomorphic) public-key encryption [9, 13], hash proof systems (HPS) [14], and non-interactive zero-knowledge (NIZK) proof systems [2]. However, due to the restriction on running resources, many important primitives remain unknown. Surprisingly, digital signature schemes are among them, although they are implied by OWFs in the classical setting.

Our goal: fine-grained secure ABEs. We focus on constructing attribute-based encryption (ABE) schemes [19] with fine-grained security, since it has many applications and implies important primitives, including digital signatures. In an ABE scheme, messages are encrypted under descriptive values x , secret keys are associated with values y , and a secret key decrypts the ciphertext if and only if $p(x, y) = 1$ for some boolean predicate p . Here the predicate p may express arbitrary access policy. This is in contrast to traditional public-key encryption (PKE) schemes without access control on data. Identity-based encryption [6, 12, 28] is a simplified version of ABE, where p is the equality predicate, and it implies signatures in a natural manner (even in the fine-grained setting).

In general, it is challenging to construct ABEs. For instance, in the classical setting, it is shown that IBEs cannot be constructed using trapdoor permutations (TDP) or CCA-secure PKE schemes in a black-box manner [7]. Moreover, many

pairing-based constructions of ABE and IBE (for instance, [5, 10]) heavily rely on the algebraic structures of pairing groups. These necessary structures are not available in fine-grained cryptography. Thus, in this paper, we will transform the state of the art of fine-grained cryptography, which only provides primitives related to TDP and CCA-secure PKE, and develop new tools to achieve our goal.

1.2 Our Contributions

We construct the *first* fine-grained secure ABE scheme. In particular, our scheme is computable in $\text{AC}^0[2]$ and secure against adversaries in NC^1 . Note that $\text{AC}^0[2] \subsetneq \text{NC}^1$ [27, 30]. Similar to several existing NC^1 fine-grained primitives [9, 13, 14], the security of our scheme is based on the same worst-case assumption $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$. This is a widely accepted, weak assumption. For simplicity, we consider fine-grained cryptography as schemes with NC^1 honest users and adversaries and security based on $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$ in the rest of this paper.

Previously, fine-grained cryptography can only achieve symmetric-key and public-key encryption and HPS. Our work enriches its available tools and brings fine-grained cryptography closer to classical cryptography in terms of functionality.

In particular, our construction is presented in a generic manner using predicate encodings [10, 32]. Hence, by suitably instantiating the underlying encoding, we directly obtain a fine-grained IBE scheme (which in turn implies a fine-grained signature scheme), fine-grained ABEs for inner-product encryption, non-zero inner-product encryption, spatial encryption, doubly spatial encryption, boolean span programs, and arithmetic span programs, and also fine-grained broadcast encryption and fuzzy IBE schemes. Prior to this work, it was unknown whether these primitives can be constructed in NC^1 based on a worst-case complexity assumption.

Finally, we use our technique to construct an efficient quasi-adaptive NIZK [23] with fine-grained security. Here “quasi-adaptive” means that common reference strings may depend on the language of the NIZK system.

1.3 Technique Overview

We borrow the frameworks of the pairing-based constructions of IBEs in [5] and ABEs in [10] to upgrade the available fine-grained techniques [1, 14, 22] in achieving our goal. At a high-level point of view, the main idea in [5, 10] is to find a suitable symmetric-key primitive and transform it to the corresponding public-key scheme using pairings and the Matrix Decisional Diffie-Hellman (MDDH) assumption [16]. More precisely, the Blazy-Kiltz-Pan (BKP) framework [5] transforms message authentication codes (MAC) to IBEs, and the Chen-Gay-Wee (CGW) framework [10] transforms predicate encodings to ABEs.

However, the goal of fine-grained cryptography is to construct schemes with mild assumptions other than the MDDH assumption. Our work develops techniques to build ABEs without pairings or the MDDH assumption, but only under

the mild assumption that $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$. For simplicity, we mostly focus on our techniques in the context of IBE here, and give some ideas about how they can be extended to construct ABEs. In this paper, we consider adaptive security where adversaries can adaptively request user secret keys and a challenge ciphertext.

The approach of BKP and its limitations in NC^1 . The “MAC \rightarrow IBE” transformation of BKP [5] is an abstraction of the Chen-Wee (CW) IBE scheme [11], and it also generalizes the “PRF \rightarrow Signature” framework by Bellare and Goldwasser (BG) [4] in the IBE context. The BKP transformation requires an “affine MAC”, namely, a MAC whose verification is done by checking a particular system of affine equations. Variables in these affine equations are included in the MAC secret key, and the (public) coefficients are derived from the message (which will be the identity of the resulting IBE scheme) to be signed. Such a MAC scheme can be constructed based on the Diffie-Hellman assumption which is generalized as the MDDH assumption.

We give some ideas about how an affine MAC can be turned into an IBE scheme. The master public key of an IBE scheme, $\text{pk} = \text{Com}(\text{sk}_{\text{MAC}})$, is a commitment of the MAC secret key, sk_{MAC} . A user secret key $\text{usk}[\text{id}]$ of an identity id consists of a BG signature, namely, a MAC tag τ_{id} on the message id and a NIZK proof of the validity of τ_{id} w.r.t. the secret key committed in pk .

Since the MAC verification consists of only affine equations, after implementing the aforementioned commitments and NIZK proofs with the (tuned) Groth-Sahai (GS) proof system [20], the BKP IBE ciphertext ct_{id} can be viewed as a randomized linear combination of pk w.r.t. id . This is the key observation of BKP. The BKP framework can be further improved and extended to construct ABEs using predicate encodings [32] as in the CGW framework [10].

The MDDH assumption and the pairing-based GS proofs are two key ingredients for the BKP framework which are not available in fine-grained cryptography. One direction to resolve this is to develop a fine-grained GS proof system, but it is not clear what the counterpart of “pairing-product equations” will be. Instead, we achieve our goal with a simpler and more direct approach.

A hard subset membership problem for NC^1 circuits. We first need to find a counterpart of the MDDH assumption in NC^1 , since the separation assumption $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$ does not directly give us tools in constructing cryptographic schemes. In the work of [1, 22], it is shown that, if $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$ holds, then the following two distributions are indistinguishable for NC^1 circuits:

$$\underbrace{\{\mathbf{M}_0 \in \{0, 1\}^{n \times n} : \mathbf{M}_0 \stackrel{\$}{\leftarrow} \text{ZeroSamp}(n)\}}_{=D_0} \quad \text{and} \quad \underbrace{\{\mathbf{M}_1 \in \{0, 1\}^{n \times n} : \mathbf{M}_1 \stackrel{\$}{\leftarrow} \text{OneSamp}(n)\}}_{=D_1}$$

where $n = n(\lambda)$ is some polynomial in security parameter λ , and the randomized sampling algorithms ZeroSamp and OneSamp output matrices with rank $n - 1$ and full rank, respectively. Concrete definitions of these algorithms are given in Sect. 2.2, and they are not relevant in this section.

This indistinguishability implies a hard subset membership problem in NC^1 implicitly given by Egashira, Wang, and Tanaka [15] for their HPS: Given a

matrix \mathbf{M} from D_0 and a random vector \mathbf{t} in two specific distributions represented by \mathbf{M} , the task of the problem is to tell whether \mathbf{t} is in the span of \mathbf{M} .

Our IBE in NC^1 . Our main technical contribution is a new approach of using the subset membership problem to transform an affine MAC to IBEs in the fine-grained setting. Our starting point is constructing a secure affine MAC in NC^1 . We prove that, if the subset membership problem is hard in NC^1 , then our MAC is secure for NC^1 adversaries.

Next, we propose a generic construction of IBE based on affine MACs, following the BKP framework. In stark contrast to the BKP, our construction does not require pairings. Essentially, we develop a Groth-Sahai-like proof system in NC^1 to prove the validity of our affine MAC. This proof system allows us to show that if our affine MAC is secure then our resulting IBE is secure in NC^1 . At the core of our proof system is a new commitment scheme in NC^1 , for which we achieve the hiding property by exploiting the concrete structure of matrices in D_0 .

We give more details about the security proof. Firstly, the zero-knowledge property allows us to generate user secret keys for adversaries without knowing the MAC secret key. Secondly, we show that if an adversary can break the adaptive security of our IBE, then we can construct a reduction to break the security of our affine MAC. This is a crucial step, and we require some extractability of the proof system to extract the MAC forgery from the IBE adversary. In the BKP framework, this extractability can be achieved by computing the inversion of some matrix $\mathbf{A} \in \mathbb{Z}_q^{k \times k}$ for some positive integer k . However, in our setting, inverting a matrix in $\{0, 1\}^{n \times n}$ is impossible, otherwise, this will lead to a distinguisher for the subset membership problem in NC^1 . Also, there is no known way to sample a matrix with its inverse efficiently [14]. To solve it, our proof system develop a new method in achieving this extractability without inverting any matrix. Our core idea is to prove that with a fresh random string $\mathbf{r} \leftarrow_{\$} \{0\} \times \{0, 1\}^{n-1}$, it is possible to extract the forgery from our NC^1 -commitments by switching the distribution of the public parameter $\mathbf{A} \in D_0$ twice (from D_0 to D_1 and then back to D_0) and changing the distribution of \mathbf{r} during the switching procedure.

Dual system methodology in NC^1 and ABE. Our techniques for IBE can also be viewed as the dual system encryption methodology [31] in NC^1 , which is an alternative interpretation of our approach. In our proof, there are two important technical steps, switching ciphertexts to invalid and randomizing MAC tags in the user secret keys. These correspond to switching ciphertexts and user secret keys from functional to semi-functional in the dual system encryption methodology [5, 10, 24, 31]. Dual system methodology is very useful in constructing predicate encryption and it was only known with pairings. Our work is for the first time implementing the dual system methodology without pairings.

Similar to the extension from BKP-IBE [5] to CGW-ABE [10], we further extend our techniques in constructing ABEs. We first use predicate encodings [10, 32] to generalize the notion of affine MAC and make it useful for constructing

ABEs. After that, we upgrade our IBE techniques, and transform the generalized affine MAC to an adaptively secure ABE in NC^1 .

More extension and open problem. We are optimistic that our approach can yield many more new public-key schemes in fine-grained cryptography. In particular, we show that our techniques can also be used to construct an efficient QA-NIZK in NC^1 with adaptive soundness in the full paper. Roughly, we use the technique for proving the hiding property of the underlying commitment scheme in our IBE scheme to achieve adaptive soundness.

Also, we are optimistic that our approach can be used to construct hierarchical IBE [18, 21]. We leave a detailed treatment of it as an open problem.

2 Preliminaries

Notations. We note that all arithmetic computations are over $GF(2)$ in this work. Namely, all arithmetic computations are performed with a modulus of 2. We write $a \stackrel{\$}{\leftarrow} \mathcal{A}(b)$ (respectively, $a = \mathcal{A}(b)$) to denote the random variable outputted by a probabilistic (respectively, deterministic) algorithm \mathcal{A} on input b . By $x \stackrel{\$}{\leftarrow} \mathcal{S}$ we denote the process of sampling an element x from a set or distribution \mathcal{S} uniformly at random. By $\mathbf{x} \in \{0, 1\}^n$ we denote a column vector with size n and by, say, $\mathbf{x} \in \{1\} \times \{0, 1\}^{n-1}$ we mean that the first element of \mathbf{x} is 1. By $[n]$ we denote the set $\{1, \dots, n\}$. By x_i (respectively, x_i) we denote the i th element of a vector \mathbf{x} (respectively, \mathbf{x}). By negl we denote an unspecified negligible function.

For a matrix $\mathbf{A} \in \{0, 1\}^{n \times t}$ with $\text{rank } t' < n$, we denote the sets $\{\mathbf{y} | \exists \mathbf{x} \text{ s.t. } \mathbf{y} = \mathbf{A}\mathbf{x}\}$ and $\{\mathbf{x} | \mathbf{A}\mathbf{x} = \mathbf{0}\}$ by $\text{Im}(\mathbf{A})$ (i.e., the span of \mathbf{A}) and $\text{Ker}(\mathbf{A})$ respectively. By $\mathbf{A}^\perp \in \{0, 1\}^{n \times (n-t')}$ we denote a matrix consisting of $n-t'$ linear independent column vectors in the kernel of \mathbf{A}^\top . Note that for any $\mathbf{y} \notin \text{Im}(\mathbf{A})$,

we have $\mathbf{y}^\top \mathbf{A}^\perp \neq \mathbf{0}$. By $(a_{ij})_{i \in [l], j \in [m]}$ we denote the matrix $\begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{l1} & \dots & a_{lm} \end{pmatrix}$. Let

$\mathbf{A} = (a_{ij})_{i \in [l], j \in [m]}$ be an $l \times m$ matrix and $\mathbf{B} = (\mathbf{B}_{ij})_{i \in [m], j \in [n]}$ be a large matrix consisting of $m \times n$ matrices \mathbf{B}_{ij} for all $i \in [m]$ and $j \in [n]$. By $h \odot \mathbf{A}$ we denote $(h \cdot a_{ij})_{i \in [l], j \in [m]}$ and by $\mathbf{A} \odot \mathbf{B}$ we denote

$$\left(\sum_{k=1}^m a_{ik} \odot \mathbf{B}_{kj} \right)_{i \in [l], j \in [n]}.$$

By \mathbf{M}_0^n , \mathbf{M}_1^n , and \mathbf{N}^n , we denote the following $n \times n$ matrices:

$$\mathbf{M}_0^n = \begin{pmatrix} 0 & \dots & 0 & 0 \\ 1 & 0 & & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}, \mathbf{M}_1^n = \begin{pmatrix} 0 & \dots & 0 & 1 \\ 1 & 0 & & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}, \mathbf{N}^n = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & 0 & \dots & 0 \\ 0 & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{pmatrix},$$

and by $\mathbf{0}$ we denote a zero vector $(0, \dots, 0)^\top$.

Games. We follow [5] to use code-based games for defining and proving security. A game G contains procedures `INIT` and `FINALIZE`, and some additional procedures P_1, \dots, P_n , which are defined in pseudo-code. All variables in a game are initialized as 0, and all sets are empty (denote by \emptyset). An adversary $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}}$ is executed in game G w.r.t. the security parameter λ (denote by G^{a_λ}) if a_λ first calls `INIT`, obtaining its output. Next, it may make arbitrary queries to P_i (according to their specification) and obtain their output. Finally, it makes one single call to `FINALIZE`(\cdot) and stops. We use $G^{a_\lambda} \Rightarrow d$ to denote that G outputs d after interacting with a_λ , and d is the output of `FINALIZE`.

2.1 Function Families

In this section, we recall the definitions of function families, NC^1 circuits, $AC^0[2]$ circuits, and $\oplus L/poly$. Note that $AC^0[2] \subsetneq NC^1$ [27, 30].

Definition 1 (Function Family). A function family is a family of (possibly randomized) functions $\mathcal{F} = \{f_\lambda\}_{\lambda \in \mathbb{N}}$, where for each λ , f_λ has a domain D_λ^f and a range R_λ^f .

Definition 2 (NC^1). The class of (non-uniform) NC^1 function families is the set of all function families $\mathcal{F} = \{f_\lambda\}_{\lambda \in \mathbb{N}}$ for which there is a polynomial $p(\cdot)$ and constant c such that for each λ , f_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth $c \log(\lambda)$, and fan-in 2 using `AND`, `OR`, and `NOT` gates.

Definition 3 ($AC^0[2]$). The class of (non-uniform) $AC^0[2]$ function families is the set of all function families $\mathcal{F} = \{f_\lambda\}_{\lambda \in \mathbb{N}}$ for which there is a polynomial $p(\cdot)$ and constant c such that for each λ , f_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth c , and unbounded fan-in using `AND`, `OR`, `NOT`, and `PARITY` gates.

One can see that multiplication of a constant number of matrices can be performed in $AC^0[2]$, since it can be done in constant depth with `PARITY` gates.

Definition 4 ($\oplus L/poly$). $\oplus L/poly$ is the set of all boolean function families $\mathcal{F} = \{f_\lambda\}_{\lambda \in \mathbb{N}}$ for which there is a constant c such that for each λ , there is a non-deterministic Turing machine \mathcal{M}_λ such that for each input x with length λ , $\mathcal{M}_\lambda(x)$ uses at most $c \log(\lambda)$ space, and $f_\lambda(x)$ is equal to the parity of the number of accepting paths of $\mathcal{M}_\lambda(x)$.

2.2 Sampling Procedure

We now recall the definitions of four sampling procedures `LSamp`, `RSamp`, `ZeroSamp`, and `OneSamp` in Fig. 1. Note that the output of `ZeroSamp`(n) is always a matrix of rank $n - 1$ and the output of `OneSamp`(n) is always a matrix of full rank [13].

We now recall several assumptions and lemmata on `ZeroSamp` and `OneSamp` given in [13].

Definition 5 (Fine-grained matrix linear assumption [13]). *There exists a polynomial $n = n(\lambda)$ in the security parameter λ such that for any family $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}}$ in NC^1 , we have*

$$\begin{aligned} & |\Pr[a_\lambda(\mathbf{M}) = 1 \mid \mathbf{M} \stackrel{\$}{\leftarrow} \text{ZeroSamp}(n)] \\ & - \Pr[a_\lambda(\mathbf{M}') = 1 \mid \mathbf{M}' \stackrel{\$}{\leftarrow} \text{OneSamp}(n)]| \leq \text{negl}(\lambda). \end{aligned}$$

Lemma 1 (Lemma 4.3 in [13]). *If $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$, then the fine-grained matrix linear assumption holds.*

<p>LSamp(n): For all $i, j \in [n]$ and $i < j$: $r_{i,j} \stackrel{\\$}{\leftarrow} \{0, 1\}$ Return</p> $\begin{pmatrix} 1 & r_{1,2} & \cdots & r_{1,n-1} & r_{1,n} \\ 0 & 1 & r_{2,3} & \cdots & r_{2,n} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & & 1 & r_{n-1,n} \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$	<p>RSamp(n): For $i = 1, \dots, n-1$ $r_i \stackrel{\\$}{\leftarrow} \{0, 1\}$ Return</p> $\begin{pmatrix} 1 & \cdots & 0 & r_1 \\ 0 & 1 & & r_2 \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & r_{n-1} \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$	<p>ZeroSamp(n): $\mathbf{R}_0 \stackrel{\\$}{\leftarrow} \text{LSamp}(n) \in \{0, 1\}^{n \times n}$ $\mathbf{R}_1 \stackrel{\\$}{\leftarrow} \text{RSamp}(n) \in \{0, 1\}^{n \times n}$ Return $\mathbf{R}_0 \mathbf{M}_0^n \mathbf{R}_1 \in \{0, 1\}^{n \times n}$</p> <p>OneSamp($n$): $\mathbf{R}_0 \stackrel{\\$}{\leftarrow} \text{LSamp}(n)$ $\mathbf{R}_1 \stackrel{\\$}{\leftarrow} \text{RSamp}(n)$ Return $\mathbf{R}_0 \mathbf{M}_1^n \mathbf{R}_1 \in \{0, 1\}^{n \times n}$</p>
--	---	--

Fig. 1. Definitions of LSamp, RSamp, ZeroSamp, and OneSamp. $n = n(\lambda)$ is a polynomial in the security parameter λ .

Remark. Notice that for any polynomial $n = n(\lambda)$, we have $\{f_n\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ iff $\{f_\lambda\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ since $O(\log(n(\lambda))) = O(\log(\lambda))$. Hence, in the above lemma, we can also set $n(\cdot)$ as an identity function, i.e., $n = \lambda$. For simplicity, in the rest of the paper, we always let $\text{ZeroSamp}(\cdot)$ and $\text{OneSamp}(\cdot)$ take as input λ .

The following lemma implies that for a matrix \mathbf{M}^\top sampled by $\text{ZeroSamp}(\lambda)$, there is a unique non-zero vector with the first (respectively, last) element being 1 in the kernel of \mathbf{M} (respectively, \mathbf{M}^\top).

Lemma 2 (Lemma 3 in [15]). *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$, it holds that $\text{Ker}(\mathbf{M}^\top) = \{\mathbf{0}, \mathbf{k}\}$ where \mathbf{k} is a vector such that $\mathbf{k} \in \{0, 1\}^{\lambda-1} \times \{1\}$.*

Lemma 3 Lemma 4 in [15]). *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$, it holds that $\text{Ker}(\mathbf{M}) = \{\mathbf{0}, \mathbf{k}\}$ where \mathbf{k} is a vector such that $\mathbf{k} \in \{1\} \times \{0, 1\}^{\lambda-1}$, i.e., there must exist $\mathbf{M}^\perp \in \{1\} \times \{0, 1\}^{\lambda-1}$.*

The following lemma indicates a simple relation between the distributions of the outputs of $\text{ZeroSamp}(\lambda)$ and $\text{OneSamp}(\lambda)$.

Lemma 4 (Lemma 7 in [15]). *For all $\lambda \in \mathbb{N}$, the distributions of $\mathbf{M} + \mathbf{N}^\lambda$ and \mathbf{M}' are identical, where $\mathbf{M}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ and $\mathbf{M}'^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$.*

We now give two lemmata showing that when sampling a random vector \mathbf{w} from $\{0, 1\}^\lambda$, the first element of \mathbf{w} does not affect the distribution of $\mathbf{M}\mathbf{w}$ for $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$.

Lemma 5 (Lemma 5 in [15]). *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$, it holds that*

$$\text{Im}(\mathbf{M}) = \{\mathbf{x} | \mathbf{w} \in \{0, 1\}^{\lambda-1}, \mathbf{x} = \mathbf{M}\mathbf{w}\} = \{\mathbf{x} | \mathbf{w} \in \{1\} \times \{0, 1\}^{\lambda-1}, \mathbf{x} = \mathbf{M}\mathbf{w}\}.$$

Lemma 6. *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$, the distributions of \mathbf{x} and \mathbf{x}' are identical, where $\mathbf{w} \stackrel{\$}{\leftarrow} \{0\} \times \{0, 1\}^{\lambda-1}$, $\mathbf{w}' \stackrel{\$}{\leftarrow} \{1\} \times \{0, 1\}^{\lambda-1}$, $\mathbf{x} = \mathbf{M}\mathbf{w}$, and $\mathbf{x}' = \mathbf{M}\mathbf{w}'$.*

Proof. According to Lemma 3, for any $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$, there exists $\mathbf{k} \in \text{Ker}(\mathbf{M})$ such that $\mathbf{k} \in \{1\} \times \{0, 1\}^{\lambda-1}$. Therefore, the distributions of $(\mathbf{w} + \mathbf{k})$, where $\mathbf{w} \stackrel{\$}{\leftarrow} \{0\} \times \{0, 1\}^{\lambda-1}$, and $\mathbf{w}' \stackrel{\$}{\leftarrow} \{1\} \times \{0, 1\}^{\lambda-1}$ are identical. Moreover, we have $\mathbf{M}\mathbf{w} = \mathbf{M}(\mathbf{w} + \mathbf{k})$. Hence, the distributions of $\mathbf{M}\mathbf{w}$ and $\mathbf{M}\mathbf{w}'$ are identical, completing the proof of Lemma 6. \square

Below we recall the a theorem implicitly given in [15] as the subset membership problem for an HPS. Roughly, it shows that for $\mathbf{M}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$, a vector sampled from the span of \mathbf{M} is indistinguishable from one sampled outside the span of \mathbf{M} for any adversary in NC^1 . We refer the reader to the full paper for the proof.

Definition 6 (Fine-grained subset membership problem [15]). *Let $\text{SY} = \{\text{SampYes}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\text{SN} = \{\text{SampNo}_\lambda\}_{\lambda \in \mathbb{N}}$ be function families described in Fig. 2. For all $\lambda \in \mathbb{N}$, all $\mathbf{M}^\top \in \text{ZeroSamp}(\lambda)$, and all $\mathbf{x} \in \text{SampNo}_\lambda(\mathbf{M})$, we have $\mathbf{x} \in \{0, 1\}^\lambda \setminus \text{Im}(\mathbf{M})$, then for $\mathbf{M}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ and any adversary $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$, we have*

$$\begin{aligned} & |\Pr[a_\lambda(\mathbf{x}) = 1 \mid \mathbf{x} \stackrel{\$}{\leftarrow} \text{SampYes}_\lambda(\mathbf{M})] \\ & \quad - \Pr[a_\lambda(\mathbf{x}) = 1 \mid \mathbf{x} \stackrel{\$}{\leftarrow} \text{SampNo}_\lambda(\mathbf{M})]| \leq \text{negl}(\lambda). \end{aligned}$$

$\text{SampYes}_\lambda(\mathbf{M} \in \{0, 1\}^{\lambda \times \lambda}):$ $\mathbf{w} \stackrel{\$}{\leftarrow} \{1\} \times \{0, 1\}^{\lambda-1}$ Return $\mathbf{x} = \mathbf{M}\mathbf{w}$	$\text{SampNo}_\lambda(\mathbf{M} \in \{0, 1\}^{\lambda \times \lambda}):$ $\mathbf{w} \stackrel{\$}{\leftarrow} \{1\} \times \{0, 1\}^{\lambda-1}$ Return $\mathbf{x} = (\mathbf{M} + \mathbf{N}^\lambda)\mathbf{w}$.
---	---

Fig. 2. Definitions of SY and SN. Note that $\text{SY}, \text{SN} \in \text{AC}^0[2]$, since they only involve operations including sampling random bits and multiplication of a matrix and a vector.

Theorem 1 ([15]). *If $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$, then the fine-grained subset membership problem (see Definition 6) holds.*

Remark. Note that the subset membership problem in [15] gives a stronger result additionally showing that the output distributions of $\text{SampYes}_\lambda(\mathbf{M})$ and $\text{SampNo}_\lambda(\mathbf{M})$ are identical to the uniform distributions over $\text{Im}(\mathbf{M})$ and $\{0, 1\}^\lambda \setminus \text{Im}(\mathbf{M})$ respectively. We only need a weak form of it in this work.

2.3 Predicate Encodings

We now recall the definition of predicate encodings. As in [10], our resulting construction of ABE is generally based on a predicate encoding. By exploiting various types of encodings, we can achieve a broad class of ABEs.

Our definitions are slightly different from the original definition in [10], in that our definition is over $GF(2)$ rather than $GF(p)$, and we require that the encodings are performed in a circuit class \mathcal{C}_1 .

Definition 7 (Predicate Encoding [10]). *Let $P = \{p_\lambda\}_{\lambda \in \mathbb{N}}$ with $p_\lambda : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ be a predicate, where \mathcal{X} and \mathcal{Y} are polynomial-sized spaces associated with λ . An \mathcal{C}_1 -predicate encoding for P is a function family $PE = \{rE_\lambda, kE_\lambda, sE_\lambda, sD_\lambda, rD_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_1$ with*

- $rE_\lambda : \mathcal{Y} \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\eta$,
- $kE_\lambda : \mathcal{Y} \times \{0, 1\} \rightarrow \{0, 1\}^\eta$,
- $sE_\lambda : \mathcal{X} \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\zeta$,
- $sD_\lambda : \mathcal{X} \times \mathcal{Y} \times \{0, 1\}^\zeta \rightarrow \{0, 1\}$,
- $rD_\lambda : \mathcal{X} \times \mathcal{Y} \times \{0, 1\}^\eta \rightarrow \{0, 1\}$,

where $\ell = \ell(\lambda)$, $\eta = \eta(\lambda)$, and $\zeta = \zeta(\lambda)$ are polynomials in λ .

Linearity is satisfied is for all $\lambda \in \mathbb{N}$ and all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $rE_\lambda(y, \cdot)$, $kE_\lambda(y, \cdot)$, $sE_\lambda(x, \cdot)$, $sD_\lambda(x, y, \cdot)$, and $rD_\lambda(x, y, \cdot)$ are $\{0, 1\}$ -linear. Namely, for any $y \in \mathcal{Y}$, any $\mathbf{w}_0, \mathbf{w}_1 \in \{0, 1\}^\ell$, and any $c \in \{0, 1\}$, we have $rE_\lambda(y, \mathbf{w}_0 + \mathbf{w}_1 \cdot c) = rE_\lambda(y, \mathbf{w}_0) + rE_\lambda(\mathbf{w}_1) \cdot c$, and the same argument can be made for kE_λ , sE_λ , sD_λ , and rD_λ .

Restricted α -reconstruction is satisfied if for all $\lambda \in \mathbb{N}$, all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $p_\lambda(x, y) = 1$, all $\mathbf{w} \in \{0, 1\}^\ell$, and all $\alpha \in \{0, 1\}$, we have

$$rD_\lambda(x, y, rE_\lambda(y, \mathbf{w})) = sD_\lambda(x, y, sE_\lambda(x, \mathbf{w})) \text{ and } rD_\lambda(x, y, kE_\lambda(y, \alpha)) = \alpha.$$

α -privacy is satisfied if for all $\lambda \in \mathbb{N}$, all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $p_\lambda(x, y) = 0$, and all $\alpha \in \{0, 1\}$, the following distributions are identical:

$$(x, y, \alpha, sE_\lambda(x, \mathbf{w}), rE_\lambda(y, \mathbf{w}) + kE_\lambda(y, \alpha)) \text{ and } (x, y, \alpha, sE_\lambda(x, \mathbf{w}), rE_\lambda(y, \mathbf{w})),$$

where $\mathbf{w} \stackrel{\$}{\leftarrow} \{0, 1\}^\ell$.

Remark on notions for predicate encodings. Similar to [10], we abuse the notion

$$rE_\lambda(x, \mathbf{W}) \text{ where } \mathbf{W} = (\mathbf{w}_{ij})_{i \in [l], j \in [m]} \text{ and } \mathbf{w}_{ij} \in \{0, 1\}^\ell$$

for all i, j to denote the matrix

$$(rE_\lambda(x, \mathbf{w}_{ij}))_{i \in [l], j \in [m]}.$$

The same argument is made for $(kE_\lambda, sE_\lambda, sD_\lambda, rD_\lambda)$.

Encoding for equality. We now give an example of predicate encoding PE_{eq} for equality P_{eq} in Fig. 3. By instantiating our ABKEM given later in Sect. 5 with this encoding, we immediately achieve an IBKEM. Linearity is straightforward. Restricted α -reconstruction follows from the fact that $u + \mathbf{x}^\top \mathbf{w} = u + \mathbf{y}^\top \mathbf{w}$

$\mathcal{X} = \{0, 1\}^n, \mathcal{Y} = \{0, 1\}^n$ $\ell = (1 + n), \eta = 1, \zeta = 1$	$\text{sE}_\lambda(\mathbf{x}, (u, \mathbf{w}^\top)^\top) = u + \mathbf{x}^\top \mathbf{w}$ $\text{rE}_\lambda(\mathbf{y}, (u, \mathbf{w}^\top)^\top) = u + \mathbf{y}^\top \mathbf{w}$ $\text{kE}_\lambda(\mathbf{y}, \alpha) = \alpha$ $\text{sD}_\lambda(\mathbf{x}, \mathbf{y}, c) = c$ $\text{rD}_\lambda(\mathbf{x}, \mathbf{y}, d) = d$
$\text{p}_\lambda(\mathbf{x}, \mathbf{y})$: Return 1 iff $\mathbf{x} = \mathbf{y}$	

Fig. 3. Definitions of $\text{P}_{\text{eq}} = \{\text{p}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\text{PE}_{\text{eq}} = \{\text{rE}_\lambda, \text{kE}_\lambda, \text{sE}_\lambda, \text{sD}_\lambda, \text{rD}_\lambda\}$.

when $\mathbf{x} = \mathbf{y}$, and α -privacy follows from the fact that $u + \mathbf{x}^\top \mathbf{w}$ and $u + \mathbf{y}^\top \mathbf{w}$ are pairwise independent if $\mathbf{x} \neq \mathbf{y}$.

2.4 Attribute-Based Key Encapsulation

We now give the definition of fine-grained ABKEM, the instantiation of which can be easily converted into ABEs by using a one-time symmetric cypher.

Definition 8 (Attribute-Based Key Encapsulation). A \mathcal{C}_1 -attribute-based key encapsulation (ABKEM) scheme for a predicate $\text{P} = \{\text{p}_\lambda\}_\lambda$ is a function family $\text{ABKEM} = \{\text{Gen}_\lambda, \text{USKGen}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_1$ with the following properties.

- Gen_λ returns the (master) public/secret key (pk, sk) . We assume that pk implicitly defines value spaces \mathcal{X} and \mathcal{Y} , a key space \mathcal{K} , and a ciphertext space \mathcal{C} .
- $\text{USKGen}_\lambda(\text{sk}, y)$ returns a user secret-key $\text{usk}[y]$ for a value $y \in \mathcal{Y}$.
- $\text{Enc}_\lambda(\text{pk}, x)$ returns a symmetric key $\text{K} \in \mathcal{K}$ together with a ciphertext $\text{ct} \in \mathcal{C}$ w.r.t. $x \in \mathcal{X}$.
- $\text{Dec}_\lambda(\text{usk}[y], x, \text{ct})$ deterministically returns a decapsulated key $\text{K} \in \mathcal{K}$ or the reject symbol \perp .

Perfect correctness is satisfied if for all $\lambda \in \mathbb{N}$, all $(\text{pk}, \text{sk}) \in \text{Gen}_\lambda$, all $y \in \mathcal{Y}$, all $x \in \mathcal{X}$, all $\text{usk}[y] \in \text{USKGen}_\lambda(\text{sk}, y)$, and all $(\text{K}, \text{ct}) \in \text{Enc}_\lambda(\text{pk}, x)$, if $\text{p}_\lambda(x, y) = 1$, we have

$$\Pr[\text{Dec}_\lambda(\text{pk}, \text{usk}[y], \text{ct}) = \text{K}] = 1.$$

The security requirement we consider is indistinguishability against chosen plaintext and attribute attacks (PR-AT-CPA) defined as follows.

Definition 9 (PR-AT-CPA Security for ABKEM). Let $k(\cdot)$ and $l(\cdot)$ be functions in λ . ABKEM is \mathcal{C}_2 - (k, l) -PR-AT-CPA secure if for any $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_2$, where a_λ is allowed to make k rounds of adaptive queries to $\text{USKGen}(\cdot)$ and each round it query l inputs, we have

$$|\Pr[\text{PR-AT-CPA}_{\text{real}}^{a_\lambda} \Rightarrow 1] - \Pr[\text{PR-AT-CPA}_{\text{rand}}^{a_\lambda} \Rightarrow 1]| \leq \text{negl}(\lambda),$$

where the experiments are defined in Fig. 4.

<p><u>INIT:</u> $(pk, sk) \stackrel{s}{\leftarrow} \text{Gen}_\lambda$ Return pk</p> <p><u>USKGEN(y):</u> // $k(\lambda) \times l(\lambda)$ queries $\mathcal{Q}_y \stackrel{s}{\leftarrow} \mathcal{Q}_y \cup \{y\}$ Return $\text{usk}[id] \stackrel{s}{\leftarrow} \text{USKGen}_\lambda(sk, y)$</p>	<p><u>ENC(x):</u> // one query $(K^*, ct^*) \stackrel{s}{\leftarrow} \text{Enc}_\lambda(pk, x)$</p> <div style="border: 1px solid black; padding: 2px; width: fit-content; margin: 2px auto;"> $K^* \stackrel{s}{\leftarrow} \mathcal{K}$ </div> Return (K^*, ct^*) <p><u>FINALIZE(β):</u> If $(p_\lambda(x, y) \neq 1$ for all $y \in \mathcal{Q}_y$, return β Else return 0</p>
---	---

Fig. 4. Security Games $\text{PR-AT-CPA}_{\text{real}}$ and $\text{PR-AT-CPA}_{\text{rand}}$ for defining PR-AT-CPA security for ABKEM. The boxed statement redefining K^* is only executed in game $\text{PR-AT-CPA}_{\text{rand}}$.

3 Generalized Affine MAC

In this section, we give the definition of generalized affine MAC, which generalizes the notion of standard affine MAC [5] by using predicate encodings, and show how to construct it in the fine-grained setting under the assumption $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$.

3.1 Definitions

The definition of generalized affine MAC is as follows.

Definition 10 (Generalized Affine MAC). Let $\text{PE} = \{\text{sE}_\lambda, \text{rE}_\lambda, \text{kE}_\lambda, \text{sD}_\lambda, \text{rD}_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_1$ be a predicate encoding for $\text{P} = \{p_\lambda\}_{\lambda \in \mathbb{N}}$, where $\text{rE}_\lambda : \mathcal{Y} \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\eta$, $\text{kE}_\lambda : \mathcal{Y} \times \{0, 1\} \rightarrow \{0, 1\}^\eta$, and $\text{sE}_\lambda : \mathcal{X} \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\zeta$.

A \mathcal{C}_1 -generalized affine message authentication code for PE is a function family $\text{MAC}_{\text{GA}} = \{\text{Gen}_{\text{MAC}_\lambda}, \text{Tag}_\lambda, \text{Ver}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_1$.

1. $\text{Gen}_{\text{MAC}_\lambda}$ returns sk_{MAC} containing $(\mathbf{B}, \mathbf{X}, x')$, where $\mathbf{B} \in \text{ZeroSamp}(\lambda)$, $\mathbf{X} \in \{0, 1\}^{\lambda \times \ell}$, and $x' \in \{0, 1\}$.
2. $\text{Tag}_\lambda(\text{sk}_{\text{MAC}}, m \in \mathcal{Y})$ returns a tag $\tau = (\mathbf{t}, \mathbf{u}) \in \{0, 1\}^\lambda \times \{0, 1\}^\eta$, computed as

$$\mathbf{t} \stackrel{s}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B}) \quad (1)$$

$$\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta. \quad (2)$$

3. $\text{Ver}_{\text{MAC}_\lambda}(\text{sk}_{\text{MAC}}, m, \tau = (\mathbf{t}, \mathbf{u}))$ verifies if Eq. (2) holds.

Correctness is satisfied if for any $\text{sk}_{\text{MAC}} \in \text{Gen}_{\text{MAC}_\lambda}$, $m \in \mathcal{Y}$, and $\tau \in \text{Tag}_\lambda(\text{sk}_{\text{MAC}}, m)$, we have $1 = \text{Ver}_{\text{MAC}_\lambda}(\text{sk}_{\text{MAC}}, m, \tau)$.

The security requirement we consider is pseudorandomness against chosen message attacks (PR-CMA) defined as follows.

<p><u>INIT:</u> $\text{sk}_{\text{MAC}} = (\mathbf{B}, \mathbf{X}, x') \xleftarrow{\\$} \text{Gen}_{\text{MAC}_\lambda}(\text{par})$ Return ε</p> <p><u>EVAL(m):</u> // $k(\lambda) \times \ell(\lambda)$ queries $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ Return $(\mathbf{t}, \mathbf{u}) \xleftarrow{\\$} \text{Tag}_\lambda(\text{sk}_{\text{MAC}}, m)$</p>	<p><u>CHAL(m*):</u> //one query $\mathbf{h}_0 = \text{sE}_\lambda(m^*, \mathbf{X}^\top) \in \{0, 1\}^{\zeta \times \lambda}$ $h_1 = x' \in \{0, 1\}$ $h_1 \xleftarrow{\\$} \{0, 1\}$ Return (h, \mathbf{h}_0, h_1)</p> <p><u>FINALIZE($\beta \in \{0, 1\}$):</u> If $p_\lambda(m^*, m) \neq 1$ for all $m \in \mathcal{Q}_m$, return β Else return 0</p>
--	--

Fig. 5. Games $\text{PR-CMA}_{\text{real}}$ and $\text{PR-CMA}_{\text{rand}}$ for defining PR-CMA security. The boxed statement redefining h_1 is only executed in game $\text{PR-CMA}_{\text{rand}}$.

Definition 11 (PR-CMA Security). Let $k = k(\lambda)$ and $l = l(\lambda)$ be polynomials in λ . MAC_{GA} is \mathcal{C}_2 - (k, l) -PR-CMA secure if for any $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_2$, where a_λ is allowed to make k rounds of adaptive queries to $\text{EVAL}(\cdot)$ and each round it queries l inputs, we have

$$\Pr[\text{PR-CMA}_{\text{real}}^{a_\lambda} \Rightarrow 1] - \Pr[\text{PR-CMA}_{\text{rand}}^{a_\lambda} \Rightarrow 1] \leq \text{negl}(\lambda),$$

where the experiments are defined in Fig. 5.

Roughly, the PR-CMA security says that in the presence of many tags and a challenge token (h, \mathbf{h}_0, h_1) , an adversary cannot tell whether the h_1 is honestly generated or randomness.

Standard Affine MAC. Let $\mathbf{X} = (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_n) \xleftarrow{\$} \{0, 1\}^{\lambda \times (n+1)}$. When $p_\lambda(\cdot)$ is an identity function, \mathbf{u} is computed as

$$u = \mathbf{x}_0^\top \mathbf{t} + \sum_{i=1}^n m_i \mathbf{x}_i^\top \mathbf{t} + x' \in \{0, 1\} \quad (3)$$

in Eq. (2), and \mathbf{h}_0 is computed as

$$\mathbf{h}_0 = h \cdot (\mathbf{x}_0^\top + \sum_{i=1}^n m_i \mathbf{x}_i^\top) \in \{0, 1\}^{1 \times \lambda} \quad (4)$$

in Fig. 5, i.e., the predicate encoding is the one for equality (see Fig. 3), the above definition becomes exactly the same as that of affine MAC given in [5] for the HPS based IBKEM, except that we only consider computations over $GF(2)$ and \mathbf{t} is sampled by SampYes_λ . We give the definition as below.

Definition 12 (Affine MAC [5]). A Generalized affine MAC for the predicate P_{eq} and encoding PE_{eq} defined as in Fig. 3 is said to be an affine MAC.

$\text{Gen}_{\text{MAC}_\lambda}(\text{par}):$ $\mathbf{B}^\top \stackrel{s}{\leftarrow} \text{ZeroSamp}(\lambda)$ $\mathbf{X} \stackrel{s}{\leftarrow} \{0, 1\}^{\lambda \times \ell}$ $x' \stackrel{s}{\leftarrow} \{0, 1\}$ Return $\text{sk}_{\text{MAC}} = (\mathbf{B}, \mathbf{X}, x')$	$\text{Tag}_\lambda(\text{sk}_{\text{MAC}}, m \in \mathcal{Y}):$ $\mathbf{t} \stackrel{s}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ Return $\tau = (\mathbf{t}, \mathbf{u})$ <hr style="border: 0.5px solid black;"/> $\text{Ver}_{\text{MAC}_\lambda}(\text{sk}_{\text{MAC}}, m \in \mathcal{Y}, \tau):$ If $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x')$ return 1 Else return 0
---	---

Fig. 6. Definition of $\text{MAC}_{\text{GA}} = \{\text{Gen}_{\text{MAC}_\lambda}, \text{Tag}_\lambda, \text{Ver}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}}$.

3.2 Construction

In this section, we give our construction of $\text{AC}^0[2]$ -generalized affine MAC based on $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$. It is a natural extension of the standard affine MAC from an HPS in [5].

Theorem 2. *If $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$ and $\text{PE} = \{\text{sE}_\lambda, \text{rE}_\lambda, \text{kE}_\lambda, \text{sD}_\lambda, \text{rD}_\lambda\}_{\lambda \in \mathbb{N}} \in \text{AC}^0[2]$ is a predicate encoding, where $\text{rE}_\lambda : \mathcal{Y} \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\eta$, $\text{kE}_\lambda : \mathcal{Y} \times \{0, 1\} \rightarrow \{0, 1\}^\eta$, and $\text{sE}_\lambda : \mathcal{X} \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\zeta$, then MAC_{GA} is an $\text{AC}^0[2]$ -generalized affine MAC that is NC^1 - (k, l) -PR-CMA secure, where k is any constant and $l = l(\lambda)$ is any polynomial in λ .*

Proof. First, we note that $(\{\text{Gen}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}}, \{\text{Tag}_\lambda\}_{\lambda \in \mathbb{N}}, \{\text{Ver}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}})$ are computable in $\text{AC}^0[2]$, since they only involve operations including sampling random bits and multiplication of a constant number of matrices, which can be done in constant depth with PARITY gates. Also, it is straightforward that MAC_{GA} satisfies correctness.

We now prove that MAC_{GA} is NC^1 - (k, l) -PR-CMA secure by defining a sequence of intermediate games as in Fig. 7.

Let $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ be any adversary against the PR-CMA-security of MAC_{GA} . Game G_0 is the real attack game. In games $\text{G}_{1,i}$, the first $i - 1$ queries to the EVAL oracle are answered with (\mathbf{t}, \mathbf{u}) , where $\mathbf{t} \stackrel{s}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})$ and \mathbf{u} contains no information on $\text{kE}_\lambda(m, x')$, and the remaining are answered as in the real scheme. To interpolate between $\text{G}_{1,i}$ and $\text{G}_{1,i+1}$, we also define $\text{G}'_{1,i}$, which answers the i -th query to EVAL by picking $\mathbf{t} \stackrel{s}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})$. By definition, we have $\text{G}_0 = \text{G}_{1,1}$.

Lemma 7. $\Pr[\text{PR-CMA}_{\text{real}}^{a_\lambda} \Rightarrow 1] = \Pr[\text{G}_0^{a_\lambda} \Rightarrow 1] = \Pr[\text{G}_{1,1}^{a_\lambda} \Rightarrow 1]$.

Lemma 8. *There exists an adversary $\mathcal{B}_{1,i} = \{b_\lambda^{1,i}\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ such that $b_\lambda^{1,i}$ breaks the fine-grained subset membership problem (see Definition 6), which holds under $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$ according to Lemma 1, with probability*

$$|\Pr[\text{G}'_{1,i}{}^{a_\lambda} \Rightarrow 1] - \Pr[\text{G}_{1,i}{}^{a_\lambda} \Rightarrow 1]|.$$

<p><u>INIT:</u> // Games G_0-G_2 $\mathbf{B}^\top \stackrel{\\$}{\leftarrow} \text{ZeroSamp}(\lambda)$, $x' \stackrel{\\$}{\leftarrow} \{0, 1\}$ For $\mathbf{X} \stackrel{\\$}{\leftarrow} \{0, 1\}^{\lambda \times \ell}$ Return ε</p> <p><u>CHAL</u>($m^* \in \mathcal{X}$): // Games G_0-$G_{1,Q+1}$, $\boxed{G_2}$ $\mathbf{h}_0 = \text{sE}_\lambda(m^*, \mathbf{X}^\top) \in \{0, 1\}^{\zeta \times \lambda}$ $h_1 = x' \in \{0, 1\}$ $\boxed{h_1 \stackrel{\\$}{\leftarrow} \{0, 1\}}$ Return (\mathbf{h}_0, h_1)</p> <p><u>FINALIZE</u>($\beta \in \{0, 1\}$): // Games G_0-G_2 If $\text{p}_\lambda(m^*, m) \neq 1$ for all $m \in \mathcal{Q}_m$ return β Else return 0</p> <p><u>EVAL</u>(m): // Game G_2 $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ $\mathbf{t} \stackrel{\\$}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) \in \{0, 1\}^\eta$ Return (\mathbf{t}, \mathbf{u})</p>	<p><u>EVAL</u>(m): // Game G_0 $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ $\mathbf{t} \stackrel{\\$}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ Return (\mathbf{t}, \mathbf{u})</p> <p><u>EVAL</u>(m): // Games $G_{1,i}$, $\boxed{G'_{1,i}}$ $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ // Let m be the c-th query ($1 \leq c \leq k \cdot l$) If $c < i$ then $\mathbf{t} \stackrel{\\$}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) \in \{0, 1\}^\eta$ If $c > i$ then $\mathbf{t} \stackrel{\\$}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ If $c = i$ then $\mathbf{t} \stackrel{\\$}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B})$ $\boxed{\mathbf{t} \stackrel{\\$}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})}$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ Return (\mathbf{t}, \mathbf{u})</p>
---	---

Fig. 7. Games G_0 , $(G_{1,i}, G'_{1,i})_{1 \leq i \leq k \cdot l}$, $G_{1,k \cdot l + 1}$, G_2 for the proof of Theorem 2.

Proof. Games $G_{1,i}$ and $G'_{1,i}$ only differ in the distribution of \mathbf{t} returned by the EVAL oracle for its i -th query. We build $b_\lambda^{1,i}$ as follows.

The distinguisher $b_\lambda^{1,i}$ runs in exactly the same way as the challenger in $G_{1,i}$ except that for its i -th query, it obtains \mathbf{t} which is sampled as $\mathbf{t} \stackrel{\$}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B})$ or $\mathbf{t} \stackrel{\$}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})$. When a_λ outputs $\beta \in \{0, 1\}$, b_λ outputs β if no m such that $\text{p}_\lambda(m^*, m) = 1$ was queried to EVAL. Otherwise, b_λ outputs 0.

Since a_λ only makes constant rounds of queries, all the operations in b_λ are performed in NC^1 . Hence, we have $\mathcal{B}_{1,i} \in \text{NC}^1$.

When \mathbf{t} is sampled as $\mathbf{t} \stackrel{\$}{\leftarrow} \text{SampYes}_\lambda(\mathbf{B})$ (respectively, $\mathbf{t} \stackrel{\$}{\leftarrow} \text{SampNo}_\lambda(\mathbf{B})$), the view of a_λ is exactly the same as its view in $G_{1,i}$ (respectively, $G'_{1,i}$). Thus the advantage of $b_\lambda^{1,i}$ in breaking the subset membership problem is $|\Pr[G_{1,i}^{a_\lambda} \Rightarrow 1] - \Pr[G'_{1,i} \Rightarrow 1]|$, completing this part of proof. \square

Lemma 9. $\Pr[G_{1,i+1}^{a_\lambda} \Rightarrow 1] = \Pr[G'_{1,i} \Rightarrow 1]$.

Proof. Let m be the i -th query to EVAL such that $\text{p}_\lambda(m^*, m) \neq 1$ and let (\mathbf{t}, \mathbf{u}) be its tag. We have $\mathbf{t} \notin \text{Im}(\mathbf{B})$ due to Lemma 1. We use an information-theoretic argument to show that in $G'_{1,i}$, \mathbf{u} does not reveal any information on x' . Information-theoretically, a_λ may learn $\mathbf{B}^\top \mathbf{X}$ from each c -th query with $c > i$. Thus, for $\mathbf{X} \stackrel{\$}{\leftarrow} \{0, 1\}^{\lambda \times \ell}$ and $\mathbf{w} \stackrel{\$}{\leftarrow} \{0, 1\}^{\ell \times 1}$, a_λ information-theoretically obtains the distribution of

$$\begin{aligned}
& \begin{pmatrix} \mathbf{X}^\top \mathbf{B} \\ \mathbf{h}_0 = h \odot \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{X}^\top) \\ \mathbf{u} = \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{X}^\top \mathbf{t}) + \mathbf{kE}_\lambda(\mathbf{m}, x') \end{pmatrix} \\
&= \begin{pmatrix} (\mathbf{X}^\top + \mathbf{wB}^{\perp\top}) \mathbf{B} \\ \mathbf{h}_0 = \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{X}^\top + \mathbf{wB}^{\perp\top}) \\ \mathbf{u} = \mathbf{rE}_\lambda(\mathbf{m}, (\mathbf{X}^\top + \mathbf{wB}^{\perp\top}) \mathbf{t}) + \mathbf{kE}_\lambda(\mathbf{m}, x') \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{X}^\top \mathbf{B} \\ \mathbf{h}_0 = \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{X}^\top) + \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{wB}^{\perp\top}) \\ \mathbf{u} = \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{X}^\top \mathbf{t}) + \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{w}) + \mathbf{kE}_\lambda(\mathbf{m}, x') \end{pmatrix} (\cdot: \mathbf{t} \notin \text{Im}(\mathbf{B})).
\end{aligned}$$

This distribution is identical to the distribution of

$$\begin{pmatrix} \mathbf{X}^\top \mathbf{B} \\ \mathbf{h}_0 = \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{X}^\top) + \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{wB}^{\perp\top}) \\ \mathbf{u} = \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{X}^\top \mathbf{t}) + \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{w}) \end{pmatrix},$$

since the distribution of

$$(\mathbf{m}^*, \mathbf{m}, x', \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{w}), \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{w}) + \mathbf{kE}_\lambda(\mathbf{m}, x'))$$

and

$$(\mathbf{m}^*, \mathbf{m}, x', \mathbf{sE}_\lambda(\mathbf{m}^*, \mathbf{w}), \mathbf{rE}_\lambda(\mathbf{m}, \mathbf{w})),$$

are identical due to the α -privacy of PE, completing this part of proof. \square

Lemma 10. $\Pr[\mathbf{G}_2^{a_\lambda} \Rightarrow 1] = \Pr[\mathbf{G}_{1,k,l+1}^{a_\lambda} \Rightarrow 1]$.

Proof. Note that a_λ can ask at most $k \cdot l$ -many EVAL queries. In both $\mathbf{G}_{1,k,l+1}$ and \mathbf{G}_2 , all the answers of EVAL are independent of x' . Hence, h_1 from $\mathbf{G}_{1,k,l+1}$ is uniform in the view of a_λ . \square

We now do all the previous steps in the reverse order as in Fig. 8. Then, by using the above arguments in a reverse order, we have the following lemma.

Lemma 11. *There exists an adversary $\mathcal{B}_2 = \{b_\lambda^2\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ such that b_λ^2 breaks the fine-grained subset membership problem with probability at least*

$$(|\Pr[\text{PR-CMA}_{\text{rand}}^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{G}_2^{a_\lambda} \Rightarrow 1]|)/(k \cdot l).$$

Putting all above together, Theorem 2 immediately follows. \square

An affine MAC. By instantiating the underlying predicate encoding in Fig. 6 with the encoding for equality (see Fig. 3), we immediately obtain an affine MAC $\text{MAC} = \{\text{Gen}_{\text{MAC}_\lambda}, \text{Tag}_\lambda, \text{Ver}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}}$ as in Fig. 9 for message space $\{0, 1\}^\ell$, which will be used to construct an IBE scheme in NC^1 later. Formally, we have the following corollary derived from Theorem 2.

Corollary 1. *If $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$, then MAC is an $\text{AC}^0[2]$ -affine MAC that is NC^1 - (k, l) -PR-CMA secure, where k is any constant and $l = l(\lambda)$ is any polynomial in λ .*

<p><u>INIT:</u> // Games H_0-H_2 $\mathbf{B}^\top \xleftarrow{\\$} \text{ZeroSamp}(\lambda); x' \xleftarrow{\\$} \{0, 1\}$ $\mathbf{X} \xleftarrow{\\$} \{0, 1\}^{\lambda \times \ell}$ Return ε</p> <p><u>CHAL(m^*):</u> // Games H_0-H_2 $\mathbf{h}_0 = \text{sE}_\lambda(m^*, \mathbf{X}^\top) \in \{0, 1\}^{\zeta \times \lambda}$ $h_1 \xleftarrow{\\$} \{0, 1\}$ Return (\mathbf{h}_0, h_1)</p> <p><u>FINALIZE($\beta \in \{0, 1\}$):</u> // Games H_0-H_2 If $p_\lambda(m^*, m) \neq 1$ for all $y \in \mathcal{Q}_m$ return β Else return 0</p> <p><u>EVAL(m):</u> // Game H_0 $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ $\mathbf{t} \xleftarrow{\\$} \text{SampNo}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) \in \{0, 1\}^\eta$ Return (\mathbf{t}, \mathbf{u})</p>	<p><u>EVAL(m):</u> // Games $H_{1,i}, \boxed{H'_{1,i}}$ $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ // Let m be the c-th query ($1 \leq c \leq k \cdot l$) If $c > i$ then $\mathbf{t} \xleftarrow{\\$} \text{SampNo}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) \in \{0, 1\}^\eta$ If $c < i$ then $\mathbf{t} \xleftarrow{\\$} \text{SampYes}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ If $c = i$ then $\mathbf{t} \xleftarrow{\\$} \text{SampNo}_\lambda(\mathbf{B})$ $\boxed{\mathbf{t} \xleftarrow{\\$} \text{SampYes}_\lambda(\mathbf{B})}$ $\mathbf{u} = \text{rE}_\lambda(m) \mathbf{X}^\top \mathbf{t} + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ Return (\mathbf{t}, \mathbf{u})</p> <p><u>EVAL(m):</u> // Game H_2 $\mathcal{Q}_m = \mathcal{Q}_m \cup \{m\}$ $\mathbf{t} \xleftarrow{\\$} \text{SampYes}_\lambda(\mathbf{B})$ $\mathbf{u} = \text{rE}_\lambda(m, \mathbf{X}^\top \mathbf{t}) + \text{kE}_\lambda(m, x') \in \{0, 1\}^\eta$ Return (\mathbf{t}, \mathbf{u})</p>
--	---

Fig. 8. Games $H_0, (H_{1,i}, H'_{1,i})_{1 \leq i \leq k \cdot l}, H_{1,k \cdot l+1}, H_2$ for the proof of Lemma 11.

<p><u>Gen$_{\text{MAC}_\lambda}(\text{par})$:</u> $\mathbf{B}^\top \xleftarrow{\\$} \text{ZeroSamp}(\lambda)$ $\mathbf{x}_0, \dots, \mathbf{x}_\ell \xleftarrow{\\$} \{0, 1\}^\lambda$ $x' \xleftarrow{\\$} \{0, 1\}$ Return $\text{sk}_{\text{MAC}} = (\mathbf{B}, \mathbf{x}_0, \dots, \mathbf{x}_\ell, x')$</p>	<p><u>Tag$_\lambda(\text{sk}_{\text{MAC}}, m \in \{0, 1\}^\ell)$:</u> $\mathbf{t} \xleftarrow{\\$} \text{SampYes}_\lambda(\mathbf{B})$ $u = (\mathbf{x}_0^\top + \sum_{i=1}^\ell m_i \cdot \mathbf{x}_i^\top) \mathbf{t} + x' \in \{0, 1\}$ Return $\tau = (\mathbf{t}, u)$</p> <p><u>Ver$_{\text{MAC}_\lambda}(\text{sk}_{\text{MAC}}, \tau, m)$:</u> If $u = (\mathbf{x}_0^\top + \sum_{i=1}^\ell m_i \cdot \mathbf{x}_i^\top) \mathbf{t} + x'$ return 1 Else return 0</p>
---	---

Fig. 9. Definition of $\text{MAC} = \{\text{Gen}_{\text{MAC}_\lambda}, \text{Tag}_\lambda, \text{Ver}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}}$.

4 Fine-Grained Secure Identity-Based Encryption

In this section, we present our fine-grained IBE scheme, which captures the core techniques of our ABE scheme given later in Sect. 5.

4.1 Definition

We now give the definition of fine-grained IBKEM, which is a special case of fine-grained ABKEM (see Definition 8) where the boolean predicate is restricted to be the equality predicate.

Definition 13 (Identity-Based Key Encapsulation). A \mathcal{C}_1 -identity key encapsulation (IBKEM) scheme is a function family $\text{IBKEM} = \{\text{Gen}_\lambda, \text{USKGen}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_1$ with the following properties.

- Gen_λ returns the (master) public/secret key (pk, sk) . We assume that pk implicitly defines an identity space \mathcal{ID} , a key space \mathcal{K} , and a ciphertext space \mathcal{C} .
- $\text{USKGen}_\lambda(\text{sk}, \text{id})$ returns a user secret-key $\text{usk}[\text{id}]$ for an identity $\text{id} \in \mathcal{ID}$.
- $\text{Enc}_\lambda(\text{pk}, \text{id})$ returns a symmetric key $K \in \mathcal{K}$ together with a ciphertext $\text{ct} \in \mathcal{C}$ w.r.t. $\text{id} \in \mathcal{ID}$.
- $\text{Dec}_\lambda(\text{usk}[\text{id}], \text{id}, \text{ct})$ deterministically returns a decapsulated key $K \in \mathcal{K}$ or the reject symbol \perp .

Perfect correctness is satisfied if for all $\lambda \in \mathbb{N}$, all $(\text{pk}, \text{sk}) \in \text{Gen}_\lambda$, all $\text{id} \in \mathcal{ID}$, all $\text{usk}[\text{id}] \in \text{USKGen}_\lambda(\text{sk}, \text{id})$, and all $(K, \text{ct}) \in \text{Enc}_\lambda(\text{pk}, \text{id})$, we have

$$\Pr[\text{Dec}_\lambda(\text{pk}, \text{usk}[\text{id}], \text{ct}) = K] = 1.$$

The security requirement we consider is indistinguishability against chosen plaintext and identity attacks (PR-ID-CPA) defined as follows.

Definition 14 (PR-ID-CPA Security for IBKEM). Let $k(\cdot)$ and $l(\cdot)$ be functions in λ . IBKEM is \mathcal{C}_2 - (k, l) -PR-ID-CPA secure if for any $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{C}_2$, where a_λ is allowed to make k rounds of adaptive queries to $\text{USKGEN}(\cdot)$ and each round it query l inputs, we have

$$|\Pr[\text{PR-ID-CPA}_{\text{real}}^{a_\lambda} \Rightarrow 1] - \Pr[\text{PR-ID-CPA}_{\text{rand}}^{a_\lambda} \Rightarrow 1]| \leq \text{negl}(\lambda),$$

where the experiments are defined in Fig. 10.

<p>Procedure INIT: $(\text{pk}, \text{sk}) \xleftarrow{\\$} \text{Gen}_\lambda$ Return pk</p> <p>Procedure USKGEN(id): // $k(\lambda) \times l(\lambda)$ queries $Q_{\text{id}} \xleftarrow{\\$} Q_{\text{id}} \cup \{\text{id}\}$ Return $\text{usk}[\text{id}] \xleftarrow{\\$} \text{USKGen}_\lambda(\text{sk}, \text{id})$</p>	<p>Procedure ENC(id*): // one query $(K^*, \text{ct}^*) \xleftarrow{\\$} \text{Enc}_\lambda(\text{pk}, \text{id}^*)$ <div style="border: 1px solid black; display: inline-block; padding: 2px 5px; margin: 5px 0;">$K^* \xleftarrow{\\$} \mathcal{K}$</div> Return (K^*, ct^*)</p> <p>Procedure FINALIZE(β): Return $(\text{id}^* \notin Q_{\text{id}}) \wedge \beta$</p>
---	--

Fig. 10. Security Games PR-ID-CPA_{real} and PR-ID-CPA_{rand} for defining PR-ID-CPA-security for IBKEM. The boxed statement redefining K^* is only executed in game PR-ID-CPA_{rand}.

4.2 Construction

Let $\text{MAC} = \{\text{Gen}_{\text{MAC}\lambda}, \text{Tag}_\lambda, \text{Ver}_{\text{MAC}\lambda}\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ be an affine MAC over $\{0, 1\}^\lambda$ with message space \mathcal{ID} in Fig. 9. Our IBKEM $\text{IBKEM} = \{\text{Gen}_\lambda, \text{USKGen}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda\}_{\lambda \in \mathbb{N}}$ for key-space $\mathcal{K} = \{0, 1\}$ and identity space $\{0, 1\}^\ell$ is defined as in Fig. 11.¹

<p><u>Gen_λ</u>: $\mathbf{A}^\top \xleftarrow{\\$} \text{ZeroSamp}(\lambda)$ $\text{sk}_{\text{MAC}} = (\mathbf{B}, \mathbf{x}_0, \dots, \mathbf{x}_\ell, x') \xleftarrow{\\$} \text{Gen}_{\text{MAC}\lambda}(\text{par})$ For $i = 0, \dots, \ell$: $\mathbf{Y}_i \xleftarrow{\\$} \{0, 1\}^{(\lambda-1) \times \lambda}$ $\mathbf{Z}_i = (\mathbf{Y}_i^\top \ \mathbf{x}_i) \mathbf{A} \in \{0, 1\}^{\lambda \times \lambda}$ $\mathbf{y}' \xleftarrow{\\$} \{0, 1\}^{\lambda-1}$ $\mathbf{z}' = (\mathbf{y}'^\top \ x') \mathbf{A} \in \{0, 1\}^{1 \times \lambda}$ $\text{pk} = (\mathbf{A}, (\mathbf{Z}_i)_{0 \leq i \leq \ell}, \mathbf{z}')$ $\text{sk} = (\text{sk}_{\text{MAC}}, (\mathbf{Y}_i)_{0 \leq i \leq \ell}, \mathbf{y}')$ Return (pk, sk)</p> <p><u>USKGen_λ(sk, id ∈ {0, 1}^ℓ):</u> $(\mathbf{t}, u) \xleftarrow{\\$} \text{Tag}_\lambda(\text{sk}_{\text{MAC}}, \text{id})$ $\mathbf{v} = \mathbf{t}^\top (\mathbf{Y}_0^\top + \sum_{i=1}^{\ell} \text{id}_i \odot \mathbf{Y}_i^\top) + \mathbf{y}'^\top \in \{0, 1\}^{1 \times (\lambda-1)}$ Return $\text{usk}[\text{id}] = (\mathbf{t}, u, \mathbf{v})$</p>	<p><u>Enc_λ(pk, id)</u>: $\mathbf{r} \xleftarrow{\\$} \{0\} \times \{0, 1\}^{\lambda-1}$ $\mathbf{c}_0 = \mathbf{A} \mathbf{r} \in \{0, 1\}^\lambda$ $\mathbf{c}_1 = (\mathbf{Z}_0 + \sum_{i=1}^{\ell} \text{id}_i \odot \mathbf{Z}_i) \mathbf{r} \in \{0, 1\}^\lambda$ $\mathbf{K} = \mathbf{z}' \cdot \mathbf{r} \in \{0, 1\}$. Return \mathbf{K} and $\text{ct} = (\mathbf{c}_0, \mathbf{c}_1)$</p> <p><u>Dec_λ(usk[id], id, ct)</u>: Parse $\text{usk}[\text{id}] = (\mathbf{t}, u, \mathbf{v})$ Parse $\text{ct} = (\mathbf{c}_0, \mathbf{c}_1) \in \{0, 1\}^\lambda \times \{0, 1\}^\lambda$ $\mathbf{K} = (\mathbf{v} u) \mathbf{c}_0 - \mathbf{t}^\top \mathbf{c}_1$ Return \mathbf{K}</p>
---	---

Fig. 11. Definition of our IBKEM $= \{\text{Gen}_\lambda, \text{USKGen}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda\}_{\lambda \in \mathbb{N}}$ with identity space $\{0, 1\}^\ell$ and key space $\{0, 1\}$. id_i denotes the i th bit of id for all $i \in [\ell]$.

Theorem 3. *Under the assumption $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$ and the NC^1 - (k, l) -PR-CMA security of MAC, where k is any constant and $l = l(\lambda)$ is any polynomial in λ , IBKEM is an $\text{AC}^0[2]$ -IBKEM that is NC^1 - (k, l) -PR-ID-CPA secure against NC^1 .*

Due to the page limit, we refer the reader to the full paper for the proof of Theorem 3.

Extension to IBKEM with large key space. The key space of the above IBKEM is $\{0, 1\}$, while by running it in parallel, we can easily extend it to an IBKEM with large key space. The resulting scheme can still be performed in $\text{AC}^0[2]$ since running in parallel does not increase the circuit depth. The same extension can be also made for our fine-grained secure ABKEM given later in Sect. 5.

Extension to QA-NIZK. Our techniques for proving the hiding property of the underlying commitment scheme in our IBKEM can also be used to construct an efficient fine-grained QA-NIZK in NC^1 with adaptive soundness. We refer the reader to the full paper for details.

¹ The IBKEM can be straightforwardly extended to one with large key space as we will discuss later in this section.

5 Fine-Grained Secure Attribute-Based Encryption

In this section, we generalize our IBE scheme as a fine-grained ABE scheme by using predicate encodings [10,32]. By instantiating the underlying encodings in different ways, we can achieve ABEs for inner product, non-zero inner product, spatial encryption, doubly spatial encryption, boolean span programs, and arithmetic span programs, and also broadcast encryption and fuzzy IBE schemes, which are computable in $\text{AC}^0[2]$ and secure against NC^1 under $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$. We refer the reader to the full paper for several instances of the encodings and also to [10] for more instances. We note that the encodings in [10] are defined over $GF(p)$, while the ours are over $GF(2)$. However, the proofs for encodings in [10] can be adopted in our case, since the linearity and α -reconstruction properties hold in $GF(p)$ also hold in $GF(2)$ and by the standard linear-independence arguments in $GF(2)$, the α -privacy also holds in our case.

Let $\text{PE} = \{\text{rE}_\lambda, \text{kE}_\lambda, \text{sE}_\lambda, \text{sD}_\lambda, \text{rD}_\lambda\}_{\lambda \in \mathbb{N}} \in \text{AC}^0[2]$ be a predicate encoding for $\text{P} = \{\text{p}_\lambda\}_{\lambda \in \mathbb{N}}$ with $\text{rE}_\lambda : \mathcal{Y} \times \{0,1\}^\ell \rightarrow \{0,1\}^\eta$, $\text{kE}_\lambda : \mathcal{Y} \times \{0,1\} \rightarrow \{0,1\}^\eta$, $\text{sE}_\lambda : \mathcal{X} \times \{0,1\}^\ell \rightarrow \{0,1\}^\zeta$, $\text{sD}_\lambda : \mathcal{X} \times \mathcal{Y} \times \{0,1\}^\zeta \rightarrow \{0,1\}$, and $\text{rD}_\lambda : \mathcal{X} \times \mathcal{Y} \times \{0,1\}^\eta \rightarrow \{0,1\}$. Let $\text{MAC}_{\text{GA}} = \{\text{Gen}_{\text{MAC}_\lambda}, \text{Tag}_\lambda, \text{Ver}_{\text{MAC}_\lambda}\}_{\lambda \in \mathbb{N}} \in \text{AC}^0[2]$ be a PE-generalized affine MAC over $\{0,1\}^\lambda$ with message space \mathcal{Y} . Our ABKEM $\text{ABKEM} = \{\text{Gen}_\lambda, \text{USKGen}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda\}_{\lambda \in \mathbb{N}}$ is defined as in Fig. 12.

<p><u>Gen$_\lambda$:</u> $\mathbf{A}^\top \stackrel{\\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ $\text{sk}_{\text{MAC}} = (\mathbf{B}, \mathbf{X}, x') \stackrel{\\$}{\leftarrow} \text{Gen}_{\text{MAC}_\lambda}(\text{par})$ For $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_\ell)$ and $i = 1, \dots, \ell$: $\mathbf{Y}_i \stackrel{\\$}{\leftarrow} \{0,1\}^{(\lambda-1) \times \lambda}$ $\mathbf{Z}_i = (\mathbf{Y}_i^\top \parallel \mathbf{x}_i) \mathbf{A} \in \{0,1\}^{\lambda \times \lambda}$ $\mathbf{y}' \stackrel{\\$}{\leftarrow} \{0,1\}^{(\lambda-1)}$ $\mathbf{z}' = (\mathbf{y}'^\top \parallel x') \mathbf{A} \in \{0,1\}^{1 \times \lambda}$ $\text{pk} = (\mathbf{A}, (\mathbf{Z}_i)_{1 \leq i \leq \ell}, \mathbf{z}')$ $\text{sk} = (\text{sk}_{\text{MAC}}, (\mathbf{Y}_i)_{1 \leq i \leq \ell}, \mathbf{y}')$ Return (pk, sk)</p> <p><u>USKGen$_\lambda$(sk, $y \in \mathcal{Y}$):</u> $(\mathbf{t}, \mathbf{u}) \stackrel{\\$}{\leftarrow} \text{Tag}_\lambda(\text{sk}_{\text{MAC}}, y)$ $\mathbf{v} = \text{rE}_\lambda(y, \begin{pmatrix} \mathbf{t}^\top \mathbf{Y}_1^\top \\ \vdots \\ \mathbf{t}^\top \mathbf{Y}_\ell^\top \end{pmatrix})$ $+ \text{kE}_\lambda(y, \mathbf{y}'^\top) \in \{0,1\}^{\eta \times (\lambda-1)}$ Return $\text{usk}[y] = (\mathbf{t}, \mathbf{u}, \mathbf{v})$</p>	<p><u>Enc$_\lambda$(pk, $x \in \mathcal{X}$):</u> $\mathbf{r} \stackrel{\\$}{\leftarrow} \{0\} \times \{0,1\}^{\lambda-1}$ $\mathbf{c}_0 = \mathbf{A}\mathbf{r} \in \{0,1\}^\lambda$ $\mathbf{C}_1 = \text{sE}_\lambda(x, \begin{pmatrix} \mathbf{r}^\top \mathbf{Z}_1^\top \\ \vdots \\ \mathbf{r}^\top \mathbf{Z}_\ell^\top \end{pmatrix}) \in \{0,1\}^{\zeta \times \lambda}$ $\mathbf{K} = \mathbf{z}' \cdot \mathbf{r} \in \{0,1\}$. Return \mathbf{K} and $\text{ct} = (\mathbf{c}_0, \mathbf{C}_1)$</p> <p><u>Dec$_\lambda$(pk, $\text{usk}[y], \text{ct}$):</u> Parse $\text{usk}[y] = (\mathbf{t}, \mathbf{u}, \mathbf{v})$ Parse $\text{ct} = (\mathbf{c}_0, \mathbf{C}_1)$ $\mathbf{K} = \text{rD}_\lambda(x, y, \mathbf{v} \parallel \mathbf{u}) \mathbf{c}_0$ $- \text{sD}_\lambda(x, y, \mathbf{C}_1 \mathbf{t}) \in \{0,1\}$ Return \mathbf{K}</p>
---	--

Fig. 12. Construction of $\text{ABKEM} = \{\text{Gen}_\lambda, \text{USKGen}_\lambda, \text{Enc}_\lambda, \text{Dec}_\lambda\}_{\lambda \in \mathbb{N}}$.

Theorem 4. *Under the assumption $\text{NC}^1 \not\subseteq \oplus\text{L}/\text{poly}$ and the NC^1 - (k, l) - sE_λ -PR-CMA-security of MAC_{GA} , where k is any constant and $l = l(\lambda)$ is any polynomial in λ , ABKEM is an $\text{AC}^0[2]$ -ABKEM that is NC^1 - (k, l) -PR-AT-CPA secure against NC^1 .*

Proof. First, we note that $\{\text{Gen}_\lambda\}_{\lambda \in \mathbb{N}}$, $\{\text{USKGen}_\lambda\}_{\lambda \in \mathbb{N}}$, $\{\text{Enc}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\{\text{Dec}_\lambda\}_{\lambda \in \mathbb{N}}$ are computable in $\text{AC}^0[2]$, since they only involve operations including multiplication of a constant number of matrices, sampling random bits, and running $\text{MAC}_{\text{GA}} \in \text{AC}^0[2]$.

By Equation (2) in Sect. 3.1, we have

$$\begin{aligned} & \text{rD}_\lambda(x, y, \mathbf{v} \parallel \mathbf{u}) \mathbf{c}_0 \\ = & \text{rD}_\lambda(x, y, \text{rE}_\lambda \left(y, \begin{pmatrix} \mathbf{t}^\top \mathbf{Y}_1^\top \\ \vdots \\ \mathbf{t}^\top \mathbf{Y}_\ell^\top \end{pmatrix} + \text{kE}_\lambda(y, \mathbf{y}^\top) \parallel \begin{pmatrix} \mathbf{t}^\top \mathbf{x}_1 \\ \vdots \\ \mathbf{t}^\top \mathbf{x}_\ell \end{pmatrix} + \text{kE}_\lambda(y, x') \right) \mathbf{A} \mathbf{r} \end{aligned}$$

and

$$\text{sD}_\lambda(x, y, \mathbf{C}_1 \mathbf{t}) = \text{sD}_\lambda(x, y, \text{sE}_\lambda \left(x, \begin{pmatrix} \mathbf{t}^\top (\mathbf{Y}_1^\top \parallel \mathbf{x}_1) \\ \vdots \\ \mathbf{t}^\top (\mathbf{Y}_\ell^\top \parallel \mathbf{x}_\ell) \end{pmatrix} \right) \mathbf{A} \mathbf{r}.$$

Then, due to restricted α -reconstruction (see Definition 7), the difference of the above equations yields $\mathbf{K} = (\mathbf{y}'^\top \parallel x') \mathbf{A} \mathbf{r} = \mathbf{z}' \cdot \mathbf{r}$, i.e., correctness is satisfied.

Let $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}}$ be any adversary against the NC^1 - (k, l) -PR-AT-CPA security of ABKEM. We now prove the NC^1 - (k, l) -PR-AT-CPA security by defining a sequence of games G_0 - G_6 as in Fig. 13. Roughly, in the first four games, we show how to extract a challenge token for MAC_{GA} from the challenge session key and ciphertext by switching the distribution of \mathbf{A} twice and changing the distribution of the randomness \mathbf{r} during the switching procedure. In the last two games, we show that the commitments \mathbf{Z}_i and \mathbf{z}' perfectly hide the secrets, and the answers of queries made by a_λ reveal no useful information other than the tags and token for MAC.

Lemma 12. $\Pr[\text{PR-AT-CPA}_{\text{real}}^{a_\lambda} \Rightarrow 1] = \Pr[\text{G}_1^{a_\lambda} \Rightarrow 1] = \Pr[\text{G}_0^{a_\lambda} \Rightarrow 1]$.

Proof. G_0 is the real attack game. In game G_1 , we change the simulation of \mathbf{c}_0^* , \mathbf{C}_1^* and \mathbf{K}^* in $\text{ENC}(x)$ by substituting \mathbf{Z}_i and \mathbf{z}' with their respective definitions and substituting \mathbf{A} with $\mathbf{A} + \mathbf{N}^\lambda$. Since we have

$$\mathbf{N}^\lambda \mathbf{r} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & 0 & \cdots & 0 \\ 0 & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} 0 \\ r_2 \\ \vdots \\ r_\lambda \end{pmatrix} = 0,$$

the view of a_λ in G_1 is identical to its view in G_0 , completing this part of proof. \square

INIT: //Games G_0 - G_1 , G_2 - G_3 , G_4 , G_5 - G_6

$\mathbf{A}^\top \xleftarrow{\$} \text{ZeroSamp}(\lambda)$, $\mathbf{A}^\top \xleftarrow{\$} \text{OneSamp}(\lambda)$, $\mathbf{A}^\top \xleftarrow{\$} \text{ZeroSamp}(\lambda)$

$\mathbf{R}_1 = \begin{pmatrix} \mathbf{I}_{\lambda-1} & \mathbf{0} \\ \tilde{\mathbf{r}}^\top & 1 \end{pmatrix}^\top \xleftarrow{\$} \text{RSamp}(\lambda)$, $\mathbf{R}_0 \xleftarrow{\$} \text{LSamp}(\lambda)$, $\mathbf{A}^\top = \mathbf{R}_0 \mathbf{M}_0^\lambda \mathbf{R}_1$

$\text{sk}_{\text{MAC}} = (\mathbf{B}, \mathbf{X}, x') \xleftarrow{\$} \text{Gen}_{\text{MAC}\lambda}(\mathcal{G})$

For $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_\ell)$ and $i = 1, \dots, \ell$:

$\mathbf{Y}_i \xleftarrow{\$} \{0, 1\}^{(\lambda-1) \times \lambda}$, $\mathbf{Z}_i = (\mathbf{Y}_i^\top \parallel \mathbf{x}_i) \mathbf{A} \in \{0, 1\}^{\lambda \times \lambda}$

$\mathbf{D}_i = \mathbf{Y}_i^\top + \mathbf{x}_i \cdot \tilde{\mathbf{r}}^\top \in \{0, 1\}^{\lambda \times (\lambda-1)}$, $\mathbf{Z}_i = (\mathbf{0} \parallel \mathbf{D}_i) \mathbf{R}_0^\top \in \{0, 1\}^{\lambda \times \lambda}$

$\mathbf{y}' \xleftarrow{\$} \{0, 1\}^{\lambda-1}$, $\mathbf{z}' = (\mathbf{y}'^\top \parallel x') \mathbf{A} \in \{0, 1\}^{1 \times \lambda}$

$\mathbf{d}' = \mathbf{y}'^\top + x' \cdot \tilde{\mathbf{r}}^\top \in \{0, 1\}^{1 \times (\lambda-1)}$, $\mathbf{z}' = (\mathbf{0} \parallel \mathbf{d}') \mathbf{R}_0^\top \in \{0, 1\}^{1 \times \lambda}$

$\text{pk} = (\mathbf{A}, (\mathbf{Z}_i)_{1 \leq i \leq \ell}, \mathbf{z}')$, $\text{sk} = (\text{sk}_{\text{MAC}}, (\mathbf{Y}_i)_{1 \leq i \leq \ell}, \mathbf{y}')$

Return pk

FINALIZE(β): //Games G_0 - G_6

If $(\text{p}_\lambda(x, y) \neq 1$ for all $y \in \mathcal{Q}_y$, return β

Else return 0

USKGEN(y): //Games G_0 - G_4 , G_5 - G_6

$\mathcal{Q}_y = \mathcal{Q}_y \cup \{y\}$, $(\mathbf{t}, \mathbf{u}) \xleftarrow{\$} \text{Tag}_\lambda(\text{sk}_{\text{MAC}}, y)$

$\mathbf{v} = \text{rE}_\lambda(y, \begin{pmatrix} \mathbf{t}^\top \mathbf{Y}_1^\top \\ \vdots \\ \mathbf{t}^\top \mathbf{Y}_\ell^\top \end{pmatrix}) + \text{kE}_\lambda(y, \mathbf{y}'^\top) \in \{0, 1\}^{\eta \times (\lambda-1)}$

$\mathbf{v} = \text{rE}_\lambda(y, (\mathbf{D}_1^\top \mathbf{t}, \dots, \mathbf{D}_\ell^\top \mathbf{t})^\top) + \text{kE}_\lambda(y, \mathbf{d}') - \mathbf{u} \cdot \tilde{\mathbf{r}}^\top \in \{0, 1\}^{\eta \times (\lambda-1)}$

$\text{usk}[y] = (\mathbf{t}, \mathbf{u}, \mathbf{v})$

Return $\text{usk}[y]$

ENC(x): //Games G_0 , G_1 - G_4 , G_3 - G_4 , G_5 , G_6

$\mathbf{r} \xleftarrow{\$} \{0\} \times \{0, 1\}^{\lambda-1}$, $\mathbf{r} \xleftarrow{\$} \{1\} \times \{0, 1\}^{\lambda-1}$

$\mathbf{c}_0^* = \mathbf{A} \mathbf{r} \in \{0, 1\}^\lambda$, $\mathbf{c}_0^* = (\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r}$

$\mathbf{C}_1^* = \text{sE}_\lambda(x, \begin{pmatrix} \mathbf{r}^\top \mathbf{Z}_1^\top \\ \vdots \\ \mathbf{r}^\top \mathbf{Z}_\ell^\top \end{pmatrix}) \in \{0, 1\}^{\zeta \cdot \lambda}$

$\mathbf{C}_1^* = \text{sE}_\lambda(x, ((\mathbf{Y}_1^\top \parallel \mathbf{x}_1)(\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r}, \dots, (\mathbf{Y}_\ell^\top \parallel \mathbf{x}_\ell)(\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r})^\top)$

$\mathbf{C}_1^* = \text{sE}_\lambda(x, (\mathbf{Z}_1 \mathbf{r}, \dots, \mathbf{Z}_\ell \mathbf{r})^\top) + \text{sE}_\lambda(x, (\mathbf{x}_1, \dots, \mathbf{x}_\ell)^\top)$

$\mathbf{K}^* = \mathbf{z}' \cdot \mathbf{r} \in \{0, 1\}$, $\mathbf{K}^* = (\mathbf{y}'^\top \parallel x')(\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r}$, $\mathbf{K}^* = \mathbf{z}' \cdot \mathbf{r} + x'$

$\mathbf{K}^* \xleftarrow{\$} \{0, 1\}$

Return \mathbf{K}^* and $\text{ct}^* = (\mathbf{c}_0^*, \mathbf{C}_1^*)$

Fig. 13. Games G_0 - G_6 for the proof of Theorem 4.

Lemma 13. *There exists an adversary $\mathcal{B}_1 = \{b_\lambda^1\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ such that b_λ^1 breaks the fine-grained matrix linear assumption (see Definition 5), which holds under $\text{NC}^1 \subsetneq \oplus\text{L}/\text{poly}$ according to Theorem 1, with advantage*

$$|\Pr[\mathbf{G}_2^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{G}_1^{a_\lambda} \Rightarrow 1]|.$$

Proof. \mathbf{G}_1 and \mathbf{G}_2 only differ in the distribution of \mathbf{A} , namely, $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ or $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$, and we build the distinguisher b_λ^1 as follows.

b_λ^1 runs in exactly the same way as the challenger of \mathbf{G}_1 except that in INIT, instead of generating \mathbf{A} by itself, it takes as input \mathbf{A}^\top generated as $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ or $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$ from its own challenger. When a_λ outputs β , b_λ^1 outputs β as well if no y such that $\mathfrak{p}_\lambda(x, y) = 1$ was queried to USKGEN. Otherwise, b_λ^1 outputs 0.

If \mathbf{A} is generated as $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ (respectively, $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$), the view of a_λ is the same as its view in \mathbf{G}_1 (respectively, \mathbf{G}_2). Hence, the probability that b_λ^1 breaks the fine-grained matrix linear assumption is

$$|\Pr[\mathbf{G}_2^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{G}_1^{a_\lambda} \Rightarrow 1]|.$$

Moreover, since a_λ only makes constant rounds of queries, all operations in b_λ^1 are performed in NC^1 . Hence, we have $\mathcal{B}_1 = \{b_\lambda^1\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$, completing this part of proof. \square

Lemma 14. $\Pr[\mathbf{G}_3^{a_\lambda} \Rightarrow 1] = \Pr[\mathbf{G}_2^{a_\lambda} \Rightarrow 1]$.

Proof. In this game, we sample \mathbf{r} in $\text{ENC}(x)$ as $\mathbf{r} \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda$ instead of $\mathbf{r} \stackrel{\$}{\leftarrow} \{0\} \times \{0, 1\}^{\lambda-1}$. According to Lemma 4, the distributions of $\mathbf{A} + \mathbf{N}^\lambda$ in both \mathbf{G}_2 and \mathbf{G}_3 are identical to that of a matrix sampled from ZeroSamp . Then this lemma follows from Lemma 6 immediately. \square

Lemma 15. *There exists an adversary $\mathcal{B}_2 = \{b_\lambda^2\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ such that b_λ^2 breaks the fine-grained matrix linear assumption with advantage*

$$|\Pr[\mathbf{G}_4^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{G}_3^{a_\lambda} \Rightarrow 1]|.$$

Proof. \mathbf{G}_1 and \mathbf{G}_2 only differ in the distribution of \mathbf{A} , namely, $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$ or $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$, and we build the distinguisher b_λ^2 against Lemma 1 as follows.

b_λ^2 runs in exactly the same way as the challenger of \mathbf{G}_3 except that in INIT, instead of generating \mathbf{A} by itself, it takes as input \mathbf{A}^\top generated as $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$ or $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$ from its own challenger. When a_λ outputs β , b_λ^2 outputs β as well if no y such that $\mathfrak{p}_\lambda(x, y) = 1$ was queried to USKGEN. Otherwise, b_λ^2 outputs 0.

If \mathbf{A} is generated as $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{OneSamp}(\lambda)$ (respectively, $\mathbf{A}^\top \stackrel{\$}{\leftarrow} \text{ZeroSamp}(\lambda)$), the view of a_λ is the same as its view in \mathbf{G}_3 (respectively, \mathbf{G}_4). Hence, the probability that b_λ^2 breaks the fine-grained matrix linear assumption is

$$|\Pr[\mathbf{G}_4^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{G}_3^{a_\lambda} \Rightarrow 1]|.$$

Moreover, since a_λ only makes constant rounds of queries, all operations in b_λ^2 are performed in NC^1 . Hence, we have $\mathcal{B}_2 = \{b_\lambda^2\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$, completing this part of proof. \square

Lemma 16. $\Pr[G_5^{a_\lambda} \Rightarrow 1] = \Pr[G_4^{a_\lambda} \Rightarrow 1]$.

Proof. In G_5 , we do not use $(\mathbf{Y}_i)_{i=1}^\ell$ and \mathbf{y}' in $\text{USKGEN}(\mathbf{y})$ or $\text{ENC}(\mathbf{x})$ any more. We give the sampling procedure for \mathbf{A} in an explicit way and change the simulation of \mathbf{Z}_i , \mathbf{z}' , \mathbf{v} , \mathbf{C}_1^* , and \mathbf{K}^* as in Fig. 13. We now show that all the changes are purely conceptual.

In G_5 , we generate \mathbf{A} by sampling $\mathbf{R}_1 = \begin{pmatrix} \mathbf{I}_{\lambda-1} & \mathbf{0} \\ \tilde{\mathbf{r}}^\top & 1 \end{pmatrix} \stackrel{s}{\leftarrow} \text{RSamp}(\lambda)$ and $\mathbf{R}_0 \stackrel{s}{\leftarrow} \text{LSamp}(\lambda)$, and setting $\mathbf{A}^\top = \mathbf{R}_0 \mathbf{M}_0^\lambda \mathbf{R}_1$. This is exactly the “zero-sampling” procedure, in which case, we have

$$\begin{aligned} \mathbf{Z}_i &= (\mathbf{Y}_i^\top \parallel \mathbf{x}_i) \mathbf{A} = (\mathbf{Y}_i^\top \parallel \mathbf{x}_i) \mathbf{R}_1^\top \mathbf{M}_0^\lambda \mathbf{R}_0^\top \\ &= (\mathbf{Y}_i^\top \parallel \mathbf{x}_i) \begin{pmatrix} \mathbf{I}_{\lambda-1} & \mathbf{0} \\ \tilde{\mathbf{r}}^\top & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & & 1 \\ 0 & & \cdots & & 0 \end{pmatrix} \mathbf{R}_0^\top \\ &= (\mathbf{Y}_i^\top + \mathbf{x}_i \cdot \tilde{\mathbf{r}}^\top \parallel \mathbf{x}_i) \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & & 1 \\ 0 & & \cdots & & 0 \end{pmatrix} \mathbf{R}_0^\top \\ &= (\mathbf{0} \parallel \mathbf{Y}_i^\top + \mathbf{x}_i \cdot \tilde{\mathbf{r}}^\top) \mathbf{R}_0^\top = (\mathbf{0} \parallel \mathbf{D}_i) \mathbf{R}_0^\top \end{aligned}$$

and

$$\begin{aligned} \mathbf{C}_1^* &= \text{sE}_\lambda(\mathbf{x}, ((\mathbf{Y}_1^\top \parallel \mathbf{x}_1)(\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r}, \dots, (\mathbf{Y}_\ell^\top \parallel \mathbf{x}_\ell)(\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r})^\top) \\ &= \text{sE}_\lambda(\mathbf{x}, (\mathbf{Z}_1 \mathbf{r} + \mathbf{x}_1, \dots, \mathbf{Z}_\ell \mathbf{r} + \mathbf{x}_\ell)^\top) \\ &= \text{sE}_\lambda(\mathbf{x}, (\mathbf{Z}_1 \mathbf{r}, \dots, \mathbf{Z}_\ell \mathbf{r})^\top) + \text{sE}_\lambda(\mathbf{x}, (\mathbf{x}_1, \dots, \mathbf{x}_\ell)^\top). \end{aligned}$$

Hence, the distributions of \mathbf{Z}_i in G_5 remain the same, and the distributions of \mathbf{z}' and \mathbf{K}^* can be analyzed in the same way. The distribution of \mathbf{v} does not change as well since

$$\begin{aligned} \mathbf{v} &= \text{rE}_\lambda(\mathbf{y}, (\mathbf{Y}_1 \mathbf{t}, \dots, \mathbf{Y}_\ell \mathbf{t})) + \text{kE}_\lambda(\mathbf{y}, \mathbf{y}'^\top) \\ &= \text{rE}_\lambda(\mathbf{y}, ((\mathbf{Y}_1^\top + \tilde{\mathbf{r}} \cdot \mathbf{x}_1^\top) \mathbf{t}, \dots, (\mathbf{Y}_\ell^\top + \tilde{\mathbf{r}} \cdot \mathbf{x}_\ell^\top) \mathbf{t})^\top) + \text{kE}_\lambda(\mathbf{y}, \mathbf{y}'^\top + \mathbf{x}' \cdot \tilde{\mathbf{r}}^\top) \\ &\quad - (\text{rE}_\lambda(\mathbf{y}, (\tilde{\mathbf{r}} \cdot \mathbf{x}_1^\top \cdot \mathbf{t}, \dots, \tilde{\mathbf{r}} \cdot \mathbf{x}_\ell^\top \cdot \mathbf{t})^\top) + \text{kE}_\lambda(\mathbf{y}, \mathbf{x}' \cdot \tilde{\mathbf{r}}^\top)) \\ &= \text{rE}_\lambda(\mathbf{y}, (\mathbf{D}_1^\top \mathbf{t}, \dots, \mathbf{D}_\ell^\top \mathbf{t})^\top) + \text{kE}_\lambda(\mathbf{y}, \mathbf{d}') - \mathbf{u} \cdot \tilde{\mathbf{r}}^\top. \end{aligned}$$

Putting all above together, Lemma 16 immediately follows. \square

Lemma 17. *There exists an adversary $\mathcal{B}_3 = \{b_\lambda^3\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ such that b_λ^3 breaks the NC^1 - (k, l) -PR-CMA security of MAC_{GA} with advantage*

$$|\Pr[\text{G}_6^{a_\lambda} \Rightarrow 1] - \Pr[\text{G}_5^{a_\lambda} \Rightarrow 1]|.$$

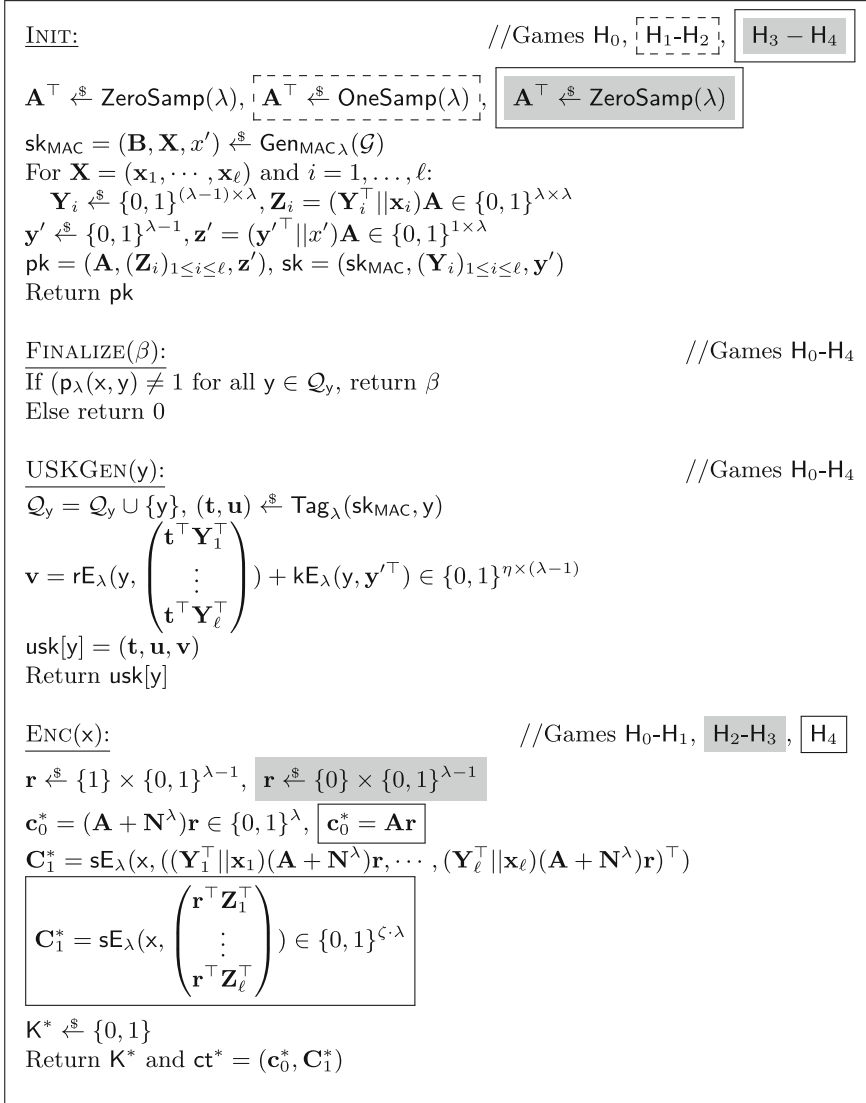
Proof. The challenger of G_6 answers the $\text{ENC}(x)$ query by choosing random K^* . We build b_λ^3 as in Fig. 14 to show that the differences between G_6 and G_5 can be bounded by its advantage of breaking the PR-CMA security of MAC_{GA} .

b_λ^3 runs in the same way as the challenger of G_5 except that it samples \mathbf{D}_i and \mathbf{d}' uniformly at random from $\{0, 1\}^{\lambda \times (\lambda-1)}$ and $\{0, 1\}^{1 \times (\lambda-1)}$ respectively. This does not change the view of a_λ since \mathbf{Y}_i and \mathbf{y}' were uniformly sampled in G_5 . Moreover, every time on receiving a query \mathbf{y} to USKGEN , b_λ^3 forwards \mathbf{y} to its evaluation oracle EVAL to obtain the answer (\mathbf{t}, \mathbf{u}) , and on receiving the query x to ENC , b_λ^3 forwards x to its challenge oracle CHAL and uses the answer (h, \mathbf{h}_0, h_1) to simulate \mathbf{r} , \mathbf{C}_1^* , and K^* as in Fig. 14. When a_λ outputs β , b_λ^3 outputs β as well if no \mathbf{y} such that $\rho_\lambda(x, \mathbf{y}) = 1$ was queried to USKGEN . Otherwise, b_λ^3 outputs 0.

<p><u>INIT:</u></p> $\mathbf{R}_1 = \begin{pmatrix} \mathbf{I}_{\lambda-1} & \mathbf{0} \\ \tilde{\mathbf{r}}^\top & 1 \end{pmatrix}^\top \stackrel{\$}{\leftarrow} \text{RSamp}(\lambda),$ $\mathbf{R}_0 \stackrel{\$}{\leftarrow} \text{LSamp}(\lambda), \mathbf{A}^\top = \mathbf{R}_0 \mathbf{M}_0^\top \mathbf{R}_1$ <p>For $i = 1, \dots, \ell$:</p> $\mathbf{D}_i \stackrel{\$}{\leftarrow} \{0, 1\}^{\lambda \times (\lambda-1)}$ $\mathbf{Z}_i = (\mathbf{0} \parallel \mathbf{D}_i) \mathbf{R}_0^\top \in \{0, 1\}^{\lambda \times \lambda}$ $\mathbf{d}' \stackrel{\$}{\leftarrow} \{0, 1\}^{1 \times (\lambda-1)}, \mathbf{z}' = (\mathbf{0} \parallel \mathbf{d}') \mathbf{R}_0^\top \in \{0, 1\}^{1 \times \lambda}$ $\text{pk} = (\mathbf{A}, (\mathbf{Z}_i)_{1 \leq i \leq \ell}, \mathbf{z}')$ <p>Return pk</p> <p><u>USKGEN(y):</u></p> $\mathcal{Q}_y = \mathcal{Q}_y \cup \{\mathbf{y}\}$ $(\mathbf{t}, \mathbf{u}) \stackrel{\$}{\leftarrow} \text{EVAL}(\mathbf{y})$ $\mathbf{v} = \mathbf{rE}_\lambda(\mathbf{y}, (\mathbf{D}_1^\top \mathbf{t}, \dots, \mathbf{D}_\ell^\top \mathbf{t})^\top) + \mathbf{kE}_\lambda(\mathbf{y}, \mathbf{d}') - \mathbf{u}$ $\tilde{\mathbf{r}}^\top \in \{0, 1\}^{\eta \times (\lambda-1)}$ $\text{usk}[\mathbf{y}] = (\mathbf{t}, \mathbf{u}, \mathbf{v})$ <p>Return usk[y]</p>	<p><u>ENC(x):</u> //one query</p> $(\mathbf{h}_0, h_1) \stackrel{\$}{\leftarrow} \text{CHAL}(x)$ $r_2, \dots, r_n \stackrel{\$}{\leftarrow} \{0, 1\}$ $\mathbf{r} = (1, r_2, \dots, r_n)^\top$ $\mathbf{c}_0^* = (\mathbf{A} + \mathbf{N}^\lambda) \mathbf{r} \in \{0, 1\}^\lambda$ $\mathbf{C}_1^* = \mathbf{sE}_\lambda(x, \begin{pmatrix} \mathbf{r}^\top \mathbf{Z}_1^\top \\ \vdots \\ \mathbf{r}^\top \mathbf{Z}_\ell^\top \end{pmatrix}) + \mathbf{h}_0 \in \{0, 1\}^{\zeta \times \lambda}$ $\text{K}^* = \mathbf{z}' \cdot \mathbf{r} + h_1 \in \{0, 1\}$ <p>Return K^* and $\text{ct}^* = (\mathbf{c}_0^*, \mathbf{C}_1^*)$</p> <p><u>FINALIZE($\beta$):</u></p> <p>If $(\rho_\lambda(x, \mathbf{y}) \neq 1$ for all $\mathbf{y} \in \mathcal{Q}_y$)</p> <p style="padding-left: 20px;">return β</p> <p>Else return 0</p>
--	---

Fig. 14. Description of $\mathcal{B}_3 = \{b_\lambda^3\}_{\lambda \in \mathbb{N}}$ (having access to the oracles INIT_{MAC} , EVAL , CHAL , $\text{FINALIZE}_{\text{MAC}}$ of the $\text{PR-CMA}_{\text{real}}/\text{PR-CMA}_{\text{rand}}$ games of Fig. 5) for the proof of Lemma 17.

If h_1 is uniform (i.e., b_λ^3 is in Game $\text{PR-CMA}_{\text{rand}}$) then the view of a_λ is identical to its view in G_6 . If h_1 is real (i.e., b_λ^3 is in Game $\text{PR-CMA}_{\text{real}}$) then the view of \mathcal{A} is identical to its view in G_5 . Hence, the advantage of b_λ^3 in breaking the PR-CMA security is

Fig. 15. Games H_0-H_4 for the proof of Theorem 4.

$$|\Pr[\mathbf{G}_6^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{G}_5^{a_\lambda} \Rightarrow 1]|.$$

Moreover, since a_λ only makes constant rounds of queries, all operations in b_λ^3 are performed in NC^1 . Hence, we have $\mathcal{B}_3 = \{b_\lambda^3\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$, completing this part of proof.

We now do all the previous steps in the reverse order as in Fig. 15. Note that the view of the adversary in H_0 (respectively, H_4) is identical to its view in \mathbf{G}_6

(respectively, PR-AT-CPA_{rand}). By using the above arguments in a reverse order, we have the following lemma.

Lemma 18. *There exists an adversary $\mathcal{B}_4 = \{b_\lambda^4\}_{\lambda \in \mathbb{N}} \in \text{NC}^1$ such that b_λ^4 breaks the fine-grained matrix linear assumption with advantage*

$$(|\Pr[\mathbf{H}_4^{a_\lambda} \Rightarrow 1] - \Pr[\mathbf{H}_0^{a_\lambda} \Rightarrow 1]|)/2.$$

□

Putting all above together, Theorem 4 immediately follows. □

Acknowledgements. We would like to thank the anonymous reviewers for their valuable comments on a previous version of this paper. Parts of Yuyu Wang’s work were supported by the National Natural Science Foundation for Young Scientists of China under Grant Number 62002049, the Fundamental Research Funds for the Central Universities under Grant Number ZYGX2020J017, and the Sichuan Science and Technology Program under Grant Numbers 2019YFG0506 and 2020YFG0292. Parts of Yu Chen’s work were supported by the National Natural Science Foundation of China under Grant Numbers 61772522 and 61932019. Parts of Jiaxin Pan’s work were supported by the Research Council of Norway under Project No. 324235.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0 . In: 45th FOCS, pp. 166–175. IEEE Computer Society Press, October 2004
2. Ball, M., Dachman-Soled, D., Kulkarni, M.: New techniques for zero-knowledge: leveraging inefficient provers to reduce assumptions, interaction, and trust. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 674–703. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_24
3. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1 . In: 18th ACM STOC, pp. 1–5. ACM Press, May 1986
4. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_19
5. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_23
6. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
7. Boneh, D., Papakonstantinou, P.A., Rackoff, C., Vahlis, Y., Waters, B.: On the impossibility of basing identity based encryption on trapdoor permutations. In: 49th FOCS, pp. 283–292. IEEE Computer Society Press, October 2008
8. Brzuska, C., Couteau, G.: Towards fine-grained one-way functions from strong average-case hardness. IACR Cryptol. ePrint Arch. **2020**, 1326 (2020)

9. Campanelli, M., Gennaro, R.: Fine-grained secure computation. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 66–97. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_3
10. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_20
11. Chen, J., Wee, H.: Fully, (Almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_25
12. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45325-3_32
13. Degwekar, A., Vaikuntanathan, V., Vasudevan, P.N.: Fine-grained cryptography. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 533–562. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_19
14. Egashira, S., Wang, Y., Tanaka, K.: Fine-grained cryptography revisited. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 637–666. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_22
15. Egashira, S., Wang, Y., Tanaka, K.: Fine-grained cryptography revisited. *J. Cryptol.* **34**(3), 23 (2021)
16. 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 (2013). https://doi.org/10.1007/978-3-642-40084-1_8
17. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
18. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_34
19. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006, pp. 89–98. ACM Press, October/November 2006, available as Cryptology ePrint Archive Report 2006/309
20. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
21. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_31
22. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS, pp. 294–304. IEEE Computer Society Press, November 2000
23. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_1

24. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_27
25. Maurer, U.M.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
26. Merkle, R.C.: Secure communications over insecure channels. *Commun. ACM* **21**(4), 294–299 (1978)
27. Razborov, A.A.: Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Acad. Sci. USSR* **41**(4) (1987)
28. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
29. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
30. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Aho, A. (ed.) 19th ACM STOC, pp. 77–82. ACM Press, May 1987
31. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36
32. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_26



Multi-input Quadratic Functional Encryption from Pairings

Shweta Agrawal^{1(✉)}, Rishab Goyal², and Junichi Tomida³

¹ IIT Madras, Chennai, India

shweta.a@cse.iitm.ac.in

² MIT, Cambridge, USA

goyal@utexas.edu

³ NTT Corporation, Tokyo, Japan

junichi.tomida.vw@hco.ntt.co.jp

Abstract. We construct the first multi-input functional encryption (MIFE) scheme for quadratic functions from pairings. Our construction supports polynomial number of users, where user i , for $i \in [n]$, encrypts input $\mathbf{x}_i \in \mathbb{Z}^m$ to obtain ciphertext CT_i , the key generator provides a key $\text{SK}_{\mathbf{c}}$ for vector $\mathbf{c} \in \mathbb{Z}^{(mn)^2}$ and decryption, given $\text{CT}_1, \dots, \text{CT}_n$ and $\text{SK}_{\mathbf{c}}$, recovers $\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$ and nothing else. We achieve indistinguishability-based (selective) security against unbounded collusions under the standard bilateral matrix Diffie-Hellman assumption. All previous MIFE schemes either support only inner products (linear functions) or rely on strong cryptographic assumptions such as indistinguishability obfuscation or multi-linear maps.

Keywords: Functional encryption · Multi-input · Quadratic functions · Pairings

1 Introduction

Functional encryption (FE) [12, 29] is a novel cryptographic paradigm that moves beyond the “all or nothing” access of traditional public key encryption and enables fine grained access to encrypted data. Concretely, an FE scheme that supports a function class \mathcal{F} allows an owner of a master secret to issue a secret key

S. Agrawal—Research supported by the DST “Swarnajayanti” fellowship, an Indo-French CEFIPRA project and the CCD Centre of Excellence. Part of the research corresponding to this work was conducted while visiting the Simons Institute for the Theory of Computing.

R. Goyal—Research supported in part by NSF CNS Award #1718161, an IBM-MIT grant, and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. Work done in part while at the Simons Institute for the Theory of Computing, supported by Simons-Berkeley research fellowship.

SK_f for a function $f \in \mathcal{F}$. Decryption of a ciphertext CT_x for a message x with SK_f yields $f(x)$ and nothing else. Functional encryption has been extensively studied in the literature, with elegant constructions supporting various function classes, achieving different notions of security and from diverse assumptions, e.g., [3, 9, 13, 19, 20].

Multi-input functional encryption (MIFE) [22] is a natural generalization of FE, which supports functions that take multiple inputs. In MIFE, multiple parties can encrypt data independently – thus, n users may encrypt their data x_1, \dots, x_n to produce ciphertexts $\text{CT}_1, \dots, \text{CT}_n$, which can be decrypted using a functional key SK_f to learn $f(x_1, \dots, x_n)$ and nothing else.

Research in MIFE has followed two broad directions. On one hand, it was shown that for general function classes (all polynomial sized circuits), FE is powerful enough to imply MIFE (albeit with exponential loss), which in turn implies the powerful notion of indistinguishability obfuscation (iO) [8, 11]. On the other hand, for restricted function classes such as constant degree polynomials, single-input schemes do not generically imply multi-input schemes and constructing multi-input schemes directly proved significantly more challenging. Intuitively, this is because in the multi-input setting, inputs x_1, \dots, x_n encrypted using independent sources of randomness must be combined in a secure way to “emulate” the single input setting where encodings of x_1, \dots, x_n may be tied together using common randomness. Nevertheless, for the inner product functionality, several novel MIFE constructions emerged based on simple, standard polynomial hardness assumptions [1, 2, 4, 6, 15, 17, 27, 30].

Beyond Inner Products. While the inner product functionality is useful for several meaningful applications (we refer the reader to [6] for a discussion), it is evidently desirable, from the viewpoint of both theory and practice, to extend the reach of MIFE from standard assumptions beyond inner products. In the single input setting, there has been significant progress in this direction. For quadratic functions, several FE schemes have been constructed from standard assumptions on pairings [9, 21, 28]¹. Indeed, from pairings, there have also been innovative constructions for “degree 2.5” FE [7], the so-called “partially hiding functional encryption” (PHFE) schemes. Intuitively, PHFE permits part of the encryptor’s input to be public and supports deeper computation on the public input as compared to the private input.

However, in the multi-input setting, constructions going beyond inner products have proved elusive. Note that unlike the single input setting, quadratic MIFE cannot be trivially constructed from inner product MIFE even with large ciphertext, since the naive idea of encrypting all quadratic monomials in advance cannot deal with quadratic terms derived from two different users. Therefore, there are currently *no* candidate constructions for MIFE supporting quadratic

¹ Note that FE for quadratic functions are trivially constructible from FE for inner products (IPFE) by linearizing and encrypting all quadratic monomials. However, FE for quadratic functions requires that the ciphertext size be linear in input length.

polynomials, from standard, polynomial hardness assumptions². This is a significant gap in our understanding of MIFE, and motivates the fundamental question:

Can we construct MIFE for quadratic functions from pairings?

1.1 Our Results

In this work, we answer the above question affirmatively and construct the first MIFE scheme for quadratic functions from pairings. In more detail, we construct n -input MIFE scheme for the function class $\mathcal{F}_{m,n}$, which is defined as follows. Each function $f \in \mathcal{F}_{m,n}$ is represented by a vector $\mathbf{c} \in \mathbb{Z}^{(mn)^2}$. For inputs $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{Z}^m$, f is defined as $f(\mathbf{x}_1, \dots, \mathbf{x}_n) := \langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$ where $\mathbf{x} = (\mathbf{x}_1 || \dots || \mathbf{x}_n)$ and \otimes denotes the Kronecker product. In a quadratic MIFE scheme for $\mathcal{F}_{m,n}$, a user can encrypt $\mathbf{x}_i \in \mathbb{Z}^m$ to CT_i for slot $i \in [n]$, a key issuer can generate a secret key SK for $\mathbf{c} \in \mathbb{Z}^{(mn)^2}$, and decryption of $\text{CT}_1, \dots, \text{CT}_n$ with SK reveals only $\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$ and nothing else.

To begin, we show that in the public key setting, quadratic MIFE can be generically obtained from public-key IPFE, which can be obtained even without pairings, in a relatively simple manner, as the case of public-key inner product MIFE [6]. Then we provide our main construction in the much more challenging secret-key setting³. Our construction relies on the bilateral matrix Diffie-Hellman assumption [18] and achieves standard indistinguishability-based (selective) security against unbounded collusions. We observe that in the symmetric key setting, selective security is the same as “semi-adaptive” [14, 23] security. Recall that in semi-adaptive security, the adversary is permitted to see the public key before committing to the challenge. In the symmetric key setting, since the “public key” is simply public parameters of the scheme, such as group description, which may always be provided to the adversary in the first step of the game, the distinction between selective and semi-adaptive is moot. Thus, our construction achieves the same level of security as single input quadratic FE [9, 21, 28].

Our construction is built using two newly introduced primitives that we call predicated IPFE and mixed-group multi-input IPFE, which we describe next. Predicated IPFE (pIPFE) is a class of attribute-based IPFE [5], but additionally with a function hiding property. In more detail, a ciphertext pCT and a secret key pSK of a pIPFE scheme pFE are associated with two vectors $\{\mathbf{x}_1, \mathbf{x}_2\}$ and $\{\mathbf{y}_1, \mathbf{y}_2\}$, respectively. Decryption of pCT with pSK reveals $\langle \mathbf{x}_2, \mathbf{y}_2 \rangle$ iff $\langle \mathbf{x}_1, \mathbf{y}_1 \rangle = 0$. Secret keys are required to hide \mathbf{y}_2 but not \mathbf{y}_1 . This scheme is the first instantiation of function-hiding attribute-based IPFE, which may be of independent interest. Mixed group multi input IPFE is similar to multi input IPFE but supports mixed

² In an exciting recent work, iO has been constructed from sub-exponential hardness of four well-founded assumptions [24]. However, this construction relies on a series of intricate, lossy reductions and is primarily a feasibility result. We will focus on the *polynomial hardness* of a well-founded problem in this work.

³ Recall that public-key MIFE does not imply secret-key MIFE. Roughly speaking, a user who has CT_1 for x_1 and SK for f of a public-key scheme is allowed to learn $f(x_1, x_2, \dots, x_n)$ for all (x_2, \dots, x_n) , since this is inherent leakage, while it is not the case in secret-key MIFE.

groups, as suggested by the name. In more detail, consider a function $f : (G_1^{m_1} \times G_2^{m_2})^n \rightarrow G_T$, specified by $([\mathbf{y}_{1,1}]_2, [\mathbf{y}_{1,2}]_1, \dots, [\mathbf{y}_{n,1}]_2, [\mathbf{y}_{n,2}]_1)$ where $\mathbf{y}_{i,1} \in \mathbb{Z}_p^{m_1}$ and $\mathbf{y}_{i,2} \in \mathbb{Z}_p^{m_2}$ and defined as $f([\mathbf{x}_{1,1}]_1, [\mathbf{x}_{1,2}]_2, \dots, ([\mathbf{x}_{n,1}]_1, [\mathbf{x}_{n,2}]_2)) := [((\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \dots, \mathbf{x}_{n,1}, \mathbf{x}_{n,2}), (\mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \dots, \mathbf{y}_{n,1}, \mathbf{y}_{n,2}))]_T$

Mixed group multi input IPFE is also required to achieve function-hiding. We provide constructions for these primitives by leveraging a (multi-input) function-hiding IPFE scheme based on pairings [4, 10, 17]. These constructions may be of independent interest.

1.2 Our Techniques

As discussed above, quadratic MIFE in the public-key setting is simple to achieve due to the leakage inherent in that setting. We formalize this in the full version of this paper. Hence, as in prior work [6], we focus on the much more challenging secret key setting. In the following, we basically use m for the vector length of each user and n for the number of slots.

Lin’s Single Key Quadratic FE. The starting point of our secret-key quadratic MIFE scheme is the secret-key quadratic FE scheme from pairings by Lin [28], which in turn builds upon the public key IPFE scheme from DDH by Abdalla *et al.* [3] (ABDP). We begin by recalling the ABDP scheme. In what follows, we let g_ℓ denote the generator of a cyclic group of order p and for matrix $\mathbf{A} = (a_{i,j})_{i,j}$, we denote $(g_\ell^{a_{i,j}})_{i,j}$ by $[\mathbf{A}]_\ell$. The ABDP scheme works as follows:

Setup(1^λ): $\mathbf{w} \leftarrow \mathbb{Z}_p^m$, PK := $[\mathbf{w}]$, MSK := \mathbf{w} .
 Enc(PK, $\mathbf{x} \in \mathbb{Z}^m$): $s \leftarrow \mathbb{Z}_p$, CT := $([s], [\mathbf{x} + s\mathbf{w}])$.
 KeyGen(MSK, $\mathbf{c} \in \mathbb{Z}^m$): SK := $-\mathbf{c}^\top \mathbf{w}$.
 Dec(CT, SK): $-\mathbf{c}^\top \mathbf{w}[s] + \mathbf{c}^\top [\mathbf{x} + s\mathbf{w}] = [(\mathbf{c}, \mathbf{x})]$.

Lin’s quadratic (secret key) FE scheme uses a clever interleaving of IPFE schemes. To compress the size of ABDP ciphertexts for quadratic terms, she leverages function-hiding IPFE, which is inherently secret-key [10]. Decryption of components in this scheme yields ciphertexts under the ABDP IPFE scheme, while secret keys of the ABDP scheme are generated using another function hiding IPFE. Finally, decryption of ABDP IPFE allows to recover the output.

In more detail, let $i\text{FE} = (i\text{Setup}, i\text{Enc}, i\text{KeyGen}, i\text{Dec})$ be a function-hiding IPFE scheme based on pairings. Note that all known function-hiding IPFE schemes based on pairings output a decryption value as an exponent of the target-group generator [10, 16, 26, 28, 31]. A simplification of her quadratic FE scheme (we omit the components of the scheme that are only required for the proof of security) is as follows:

Setup(1^λ): $\mathbf{w} = (w_1, \dots, w_m)$, $\tilde{\mathbf{w}} = (\tilde{w}_1, \dots, \tilde{w}_m) \leftarrow \mathbb{Z}_p^m$, $i\text{MSK}' \leftarrow i\text{Setup}(1^\lambda)$
 MSK := $(i\text{MSK}', \mathbf{w}, \tilde{\mathbf{w}})$.
 Enc(MSK, $\mathbf{x} \in \mathbb{Z}^m$): $s \leftarrow \mathbb{Z}_p$, $i\text{CT}' \leftarrow i\text{Enc}(i\text{MSK}', s)$, $i\text{MSK} \leftarrow i\text{Setup}(1^\lambda)$
 $i\text{CT}_i \leftarrow i\text{Enc}(i\text{MSK}, (x_i, w_i))$, $i\text{SK}_i \leftarrow i\text{KeyGen}(i\text{MSK}, (x_i, s\tilde{w}_i))$.
 CT := $(i\text{CT}', \{i\text{CT}_i, i\text{SK}_i\}_{i \in [m]})$.

KeyGen(MSK, $\mathbf{c} = \{c_{i,j}\}_{i,j \in [m]} \in \mathbb{Z}^{m^2}$):

SK := iSK' \leftarrow iKeyGen(MSK', $-\mathbf{c}^\top(\mathbf{w} \otimes \tilde{\mathbf{w}})$).

Dec(CT, SK): iDec(iCT', iSK') + $\sum_{i,j \in [m]} c_{i,j}$ iDec(iCT_{*i*}, iSK_{*j*}) = $[\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle]_T$.

To decrypt, we compute $\text{iDec}(\text{iCT}_i, \text{iSK}_j) = [x_i x_j + s w_i \tilde{w}_j]_T$, which can be seen as the (i, j) -th element of the ABDP ciphertext $[\mathbf{x} \otimes \mathbf{x} + s \mathbf{w} \otimes \tilde{\mathbf{w}}]_T$, and $\text{iDec}(\text{iCT}', \text{iSK}') = [-s \mathbf{c}^\top(\mathbf{w} \otimes \tilde{\mathbf{w}})]_T$, where $-\mathbf{c}^\top(\mathbf{w} \otimes \tilde{\mathbf{w}})$ is an ABDP secret key for \mathbf{c} . The function-hiding property of iFE guarantees that iSK hides x_i . Since $\mathbf{w} \otimes \tilde{\mathbf{w}}$ only appears on the exponent, one can argue that it is computationally indistinguishable from random in the security proof using the SXDH assumption.

IP-MIFE instead of IPFE. To generalize the above scheme to the multi-input setting, our first attempt is to modify Lin's scheme so that decryption of the function hiding IPFE scheme generates ciphertexts of a *multi-input* IPFE (IP-MIFE) scheme [4] (ACFGU) instead of a single input IPFE scheme (ABDP). Intuitively, the reason for using IP-MIFE instead of IPFE is to deal with multiple independent randomnesses derived from different users, which inherently come in when generating the IPFE ciphertext elements for quadratic terms. Now, we may hope that the key generator can provide a secret key matching the ACFGU scheme so that decryption of ciphertexts of the ACFGU scheme yields the desired result. Fortunately, the ACFGU scheme does not use pairings, so this basic template does not seem impossible. However, this starting point idea runs into several hurdles as we discuss below.

Let us recall the n -input ACFGU scheme:

Setup(1^λ): MSK := $\mathbf{w}_1, \dots, \mathbf{w}_n, \mathbf{u}_1, \dots, \mathbf{u}_n \leftarrow \mathbb{Z}_p^m$.

Enc(MSK, $i, \mathbf{x}_i \in \mathbb{Z}^m$): $s_i \leftarrow \mathbb{Z}_p$, CT_{*i*} := $([s_i], [\mathbf{x}_i + s_i \mathbf{w}_i + \mathbf{u}_i])$.

KeyGen(MSK, $(\mathbf{c}_1, \dots, \mathbf{c}_n) \in \mathbb{Z}^{mn}$): SK := $(-\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{u}_i \rangle, \{-\mathbf{c}_i^\top \mathbf{w}_i\}_{i \in [n]})$.

Dec(CT₁, ..., CT_{*n*}, SK):

$$\sum_{i \in [n]} (-\mathbf{c}_i^\top \mathbf{w}_i [s_i] + \mathbf{c}_i^\top [\mathbf{x}_i + s_i \mathbf{w}_i + \mathbf{u}_i]) - [\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{u}_i \rangle] = [\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{x}_i \rangle].$$

For intuition, we note that the ACFGU scheme may be thought of as running n instances of the ABDP scheme, where each ABDP decryption outputs the i^{th} inner product $\langle \mathbf{c}_i, \mathbf{x}_i \rangle$. Revealing each partial inner product $\langle \mathbf{c}_i, \mathbf{x}_i \rangle$ would leak too much information, so these partial decryptions are masked using $\langle \mathbf{c}_i, \mathbf{u}_i \rangle$ – this creates an extra term $\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{u}_i \rangle$ during decryption, which, fortunately may be computed by the key generator and is compensated for by subtraction.

A First Candidate. Armed with these ideas, we construct a first candidate quadratic MIFE qFE = (qSetup, qEnc, qKeyGen, qDec) as follows. For ease of exposition, we assume below that the dimension of each user's input vector m is set to 1.

qSetup(1^λ): iMSK, iMSK' \leftarrow iSetup(1^λ), $w_i, \tilde{w}_i, u_i, \tilde{u}_i \leftarrow \mathbb{Z}_p$

qMSK := (iMSK, iMSK', $\{w_i, \tilde{w}_i, u_i, \tilde{u}_i\}_{i \in [n]}$).

qEnc(qMSK, $i, x_i \in \mathbb{Z}$): $s_i, \tilde{s}_i \leftarrow \mathbb{Z}_p$

iCT'_{*i*} \leftarrow iEnc(iMSK', s_i), iSK'_{*i*} \leftarrow iKeyGen(iMSK', \tilde{s}_i)

iCT_{*i*} \leftarrow iEnc(iMSK, $(x_i, s_i w_i, u_i)$), iSK_{*i*} \leftarrow iKeyGen(iMSK, $(x_i, \tilde{s}_i \tilde{w}_i, \tilde{u}_i)$)

qCT_{*i*} := (iCT'_{*i*}, iSK'_{*i*}, iCT_{*i*}, iSK_{*i*}).

$\text{qKeyGen}(\text{MSK}, \mathbf{c} = \{c_{i,j}\}_{i,j \in [n]}):$

$$\text{qSK} := ([-\sum_{i,j \in [n]} c_{i,j} u_i \tilde{u}_j]_T, \{-c_{i,j} w_i \tilde{w}_j\}_{i,j \in [n]}).$$

$\text{qDec}(\text{qCT}_1, \dots, \text{qCT}_n, \text{qSK}):$

$$\begin{aligned} & - \sum_{i,j \in [n]} c_{i,j} w_i \tilde{w}_j \text{iDec}(\text{iCT}'_i, \text{iSK}'_j) + \sum_{i,j \in [n]} c_{i,j} \text{iDec}(\text{iCT}_i, \text{iSK}_j) \\ & - [\sum_{i,j \in [n]} c_{i,j} u_i \tilde{u}_j]_T = [(\mathbf{c}, \mathbf{x} \otimes \mathbf{x})]_T \end{aligned}$$

Observe that $\{\text{iCT}_i, \text{iSK}_i\}_{i \in [n]}$ yield $\{[x_i x_j + s_i \tilde{s}_j w_i \tilde{w}_j + u_i \tilde{u}_j]_T\}_{i,j \in [n]}$ in decryption, which can be seen as ciphertexts of the n^2 -input ACFGU scheme. We also remark that we decompose the ACFGU ciphertext into ciphertexts and secret keys of function-hiding IPFE so as to allow decryptors to generate ACFGU ciphertext elements for quadratic terms derived from two different users. This is in contrast to Lin's quadratic FE scheme, which uses function-hiding IPFE to compress the ciphertext size.

However, this scheme is not secure and leaks unnecessary information to the decryptor. The problem stems for the fact that the candidate scheme allows two types of mix-and-match attacks where an adversary can simultaneously use two different ciphertexts with the same index (slot) for decryption. In more detail, the adversary can learn the following information using the current scheme. Below, the superscript denotes the ciphertext index and subscript denotes the slot in a given ciphertext – thus, qCT_i^1 denotes the 1st ciphertext for the i^{th} slot (recall there can be multiple ciphertexts in a given slot).

1. *Attack 1:* For iCT_i^1 in qCT_i^1 and iSK_i^2 in qCT_i^2 , we have that $\text{iDec}(\text{iCT}_i^1, \text{iSK}_i^2)$ is a valid ACFGU ciphertext and usable for the ACFGU decryption with qSK . This is problematic because it permits combining components from *different ciphertexts* qCT_i^1 and qCT_i^2 for the same slot i , which does not correspond to a valid combination. Recall that in an MIFE scheme, a ciphertext in slot i may be combined with multiple ciphertexts in slot $j \neq i$ but not with other ciphertexts in slot i . However, ciphertext *components* iCT_i^1 and iSK_i^1 from the same ciphertext and in the same slot i are allowed to be combined. Thus, to prevent this attack, we need to enforce that ciphertext components can be combined only when they come either from different slots or the same qCT_i .
2. *Attack 2:* Let $i_1 \neq i_2$. For $\{\text{iCT}_{i_1}^1, \text{iSK}_{i_1}^1\}$ in $\text{qCT}_{i_1}^1$, $\{\text{iCT}_{i_2}^1, \text{iSK}_{i_2}^1\}$ in $\text{qCT}_{i_2}^1$ and $\text{iSK}_{i_2}^2$ in $\text{qCT}_{i_2}^2$, we have that $\text{iDec}(\text{iCT}_{i_1}^1, \text{iSK}_{i_1}^1)$, $\text{iDec}(\text{iCT}_{i_1}^1, \text{iSK}_{i_2}^2)$ and $\text{iDec}(\text{iCT}_{i_2}^1, \text{iSK}_{i_2}^1)$ are valid ACFGU ciphertexts and usable for the decryption with qSK . This decryption leads to an *inconsistency* attack, where an adversary can compute a function over multiple ciphertexts for a given slot. As an example, let us consider the case where a decryptor has ciphertexts for (scalar) elements x_1^1, x_2^1, x_2^2 and a secret key for quadratic function $f = (c_{1,1}, c_{1,2}, c_{2,2})$ (w.l.o.g., we can assume $c_{2,1} = 0$). Now, the only valid function evaluations that an adversary should learn are

$$c_{1,1} x_1^1 x_1^1 + c_{1,2} x_1^1 x_2^1 + c_{2,2} x_2^1 x_2^1, \quad \text{and} \quad c_{1,1} x_1^1 x_1^1 + c_{1,2} x_1^1 x_2^2 + c_{2,2} x_2^2 x_2^2$$

However, the above leakage enables the adversary to additionally learn, e.g.,

$$c_{1,1} x_1^1 x_1^1 + c_{1,2} x_1^1 x_2^2 + c_{2,2} x_2^1 x_2^1$$

The above uses two different inputs (underlined) for the second slot for the same function evaluation, which is invalid.

More generally, valid combinations correspond to the set of superscripts (in red) $(1, 1), (1, 1), (1, 1)$ and $(1, 1), (1, 2), (2, 2)$. However, the adversary can learn function evaluations corresponding to $(1, 1), (1, r), (s, t)$ for any $r, s, t \in [2]$ in the current candidate scheme.

Thus, both attacks leverage the decomposable structure of the quadratic ciphertext to mix and match invalid components to obtain leakage. While both attacks have the similarity that they combine different ciphertexts for the same slot in a given evaluation, the technical treatment to handle them needs to differ. This is because to address the first attack, we must prevent the attacker from combining $(1, 1), (1, r), (s, t)$ for $s \neq t$ while for the second, we must prevent the same for $r \neq t$. Intuitively, r and t are the indices related to the ciphertexts of iFE while s is the index related to the secret keys of iFE, and thus prohibiting the case of $s \neq t$ and that of $r \neq t$ are essentially different things, which must be handled separately. Next, we describe how each of these attacks may be prevented.

Preventing Attack 1. Recall that Lin’s quadratic FE scheme does not allow attack 1 since the encryption algorithm generates a new iMSK for each ciphertext. On the other hand, our candidate uses the same iMSK for all ciphertexts so that decryptors can generate ACFGU ciphertext elements for quadratic terms from two different users. To prevent this attack, we need a function-hiding IPFE scheme where iCT is decryptable with iSK *if and only if they come from either different slots or the same* qCT_i . Thus, we need to extend the functionality of function-hiding IPFE to check the above condition prior to computation. Although this primitive is reminiscent of “attribute-based IPFE” [5], we also need the function-hiding property which has not been considered in prior works.

To address this need, we define and construct a function-hiding “predicated IPFE” (pIPFE), which can be seen as a combination of inner product encryption [25] and IPFE. Informally, a ciphertext pCT and a secret key pSK of a pIPFE scheme pFE are associated with two vectors $\{\mathbf{x}_1, \mathbf{x}_2\}$ and $\{\mathbf{y}_1, \mathbf{y}_2\}$, respectively. Here, the secret key must hide \mathbf{y}_2 but do not \mathbf{y}_1 . Decryption of pCT with pSK reveals $\langle \mathbf{x}_2, \mathbf{y}_2 \rangle$ iff $\langle \mathbf{x}_1, \mathbf{y}_1 \rangle = 0$.

To see how function-hiding predicated IPFE yields the desired functionality, let us set $\mathbf{x}_1 = (0^{2(i-1)}, 1, L, 0^{2(n-i)})$, $\mathbf{y}_1 = (0^{2(i-1)}, L, -1, 0^{2(n-i)})$ where $L \in \mathbb{Z}_p$ is sampled randomly for each encryption, and $i \in [n]$. Let (i_1, L_1) (resp. (i_2, L_2)) be a pair of a slot index and random element of \mathbf{x}_1 (resp. \mathbf{y}_1). It is easy to see that $\langle \mathbf{x}_1, \mathbf{y}_1 \rangle = 0$ iff $i_1 \neq i_2$ or $L_1 = L_2$. Since L is chosen from an exponentially large space, we have that $L_1 \neq L_2$ with overwhelming probability. We construct a function-hiding predicated IPFE scheme pFE from a function-hiding IPFE scheme iFE in a generic way. Please see Sect. 3 for details.

Preventing Attack 2. Attack 2 is much more tricky to handle. A problematic aspect of this attack is the fact that $\text{iDec}(\text{iCT}_{i_1}^1, \text{iSK}_{i_1}^1)$ and $\text{iDec}(\text{iCT}_{i_2}^1, \text{iSK}_{i_2}^1)$ are necessary for decryption of ciphertexts $\text{qCT}_{i_1}^1, \text{qCT}_{i_2}^1$ respectively, and $\text{iDec}(\text{iCT}_{i_2}^2, \text{iSK}_{i_1}^1)$ is necessary for combined decryption of the pair $\text{qCT}_{i_1}^1, \text{qCT}_{i_2}^2$.

However, they leak inappropriate information if both of them are used in decryption simultaneously. Thus, we cannot solve the problem by building in some sort of access control into iFE decryption as in the case of attack 1.

Our solution is to bind ACFGU ciphertexts generated from the iFE decryption with common random elements. That is, iCT_i in qCT_i is changed to encryption of $(x_i, s_i w_i, u_i, t_i v_i)$, and iSK_i is changed to a secret key of $(x_i, \tilde{s}_i \tilde{w}_i, r_i \tilde{u}_i, \tilde{v}_i)$ where v_i, \tilde{v}_i are new elements in $qMSK$ and r_i, t_i are the common random elements for binding ACFGU ciphertexts, which are chosen by $qEnc$. Then, decryption with $\{iCT_i, iSK_i\}_{i \in [n]}$ yields $\{[x_i x_j + s_i \tilde{s}_j w_i \tilde{w}_j + r_j u_i \tilde{u}_j + t_i v_i \tilde{v}_j]_T\}_{i, j \in [n]}$.

According to the change of iCT, iSK , the first element of an ACFGU secret key should be modified as $qSK_1 = [-\sum_{i, j \in [n]} c_{i, j} (r_j u_i \tilde{u}_j + t_i v_i \tilde{v}_j)]_T$. By this construction, we cannot simultaneously use $iDec(iCT_{i_1}^1, iSK_{i_1}^1)$, $iDec(iCT_{i_2}^1, iSK_{i_2}^1)$ and $iDec(iCT_{i_2}^2, iSK_{i_1}^1)$ for ACFGU decryption. Intuitively, qSK_1 must involve $t_{i_2}^1$ and $t_{i_2}^2$ (randomnesses used in $iCT_{i_2}^1$ and $iCT_{i_2}^2$, respectively) to decrypt the ACFGU ciphertexts generated from $iDec(iCT_{i_1}^1, iSK_{i_1}^1)$, $iDec(iCT_{i_2}^1, iSK_{i_2}^1)$ and $iDec(iCT_{i_2}^2, iSK_{i_1}^1)$ together, but in fact qSK_1 can involve only one of $t_{i_2}^1$ and $t_{i_2}^2$.

How to Generate the Modified Secret Key. The last challenge is how to generate the modified secret key. It is obvious that $qKeyGen$ cannot generate the modified key since it contains random elements r_i, t_i used in ciphertexts. We solve the problem by employing an additional function-hiding IP-MIFE scheme, denoted by $miFE$, into the candidate scheme. That is, $qEnc$ additionally generates an IP-MIFE ciphertext $miCT_i$ for (r_i, t_i) , and $qKeyGen$ generates an IP-MIFE secret key $miSK$ for $\{(\sum_{j \in [n]} c_{j, i} u_j \tilde{u}_i, \sum_{j \in [n]} c_{i, j} v_i \tilde{v}_j)\}_{i \in [n]}$. Then, a decryptor can generate the secret-key element $-\sum_{i, j \in [n]} c_{i, j} (r_j u_i \tilde{u}_j + t_i v_i \tilde{v}_j)$ from $miCT_1, \dots, miCT_n, miSK$ without knowing unnecessary information. This technique is similar to Gay's technique in [21], which uses (partially) function-hiding IPFE to generate a "decryption key" consisting of both elements inherently derived from a ciphertext and a secret key. Note that our actual scheme needs mixed-group multi-input IPFE instead of IP-MIFE, which we construct in Sect. 4.

Putting it all Together. Putting together the ideas discussed above, we now present a second version of our scheme.

$qSetup(1^\lambda)$: $iMSK' \leftarrow iSetup(1^\lambda)$, $pMSK \leftarrow pSetup(1^\lambda)$, $miMSK \leftarrow miSetup(1^\lambda)$
 $w_i, \tilde{w}_i, u_i, \tilde{u}_i, v_i, \tilde{v}_i \leftarrow \mathbb{Z}_p$
 $qMSK := (iMSK', pMSK, miMSK, \{w_i, \tilde{w}_i, u_i, \tilde{u}_i, v_i, \tilde{v}_i\}_{i \in [n]})$.
 $qEnc(qMSK, i, x_i \in \mathbb{Z})$: $s_i, \tilde{s}_i, r_i, t_i, L \leftarrow \mathbb{Z}_p$, $\ell_1 = (0^{2(i-1)}, 1, L, 0^{2(n-i)})$
 $\ell_2 = (0^{2(i-1)}, L, -1, 0^{2(n-i)})$, $iCT'_i \leftarrow iEnc(iMSK', s_i)$, $iSK'_i \leftarrow iKeyGen(iMSK', \tilde{s}_i)$
 $pCT_i \leftarrow pEnc(pMSK, \ell_1, (x_i, s_i w_i, r_i u_i, v_i))$
 $pSK_i \leftarrow pKeyGen(pMSK, \ell_2, (x_i, \tilde{s}_i \tilde{w}_i, \tilde{u}_i, t_i \tilde{v}_i))$
 $miCT_i \leftarrow miEnc(miMSK, (r_i, t_i))$, $qCT_i := (iCT'_i, iSK'_i, pCT_i, pSK_i, miCT_i)$.
 $qKeyGen(MSK, \mathbf{c} = \{c_{i, j}\}_{i, j \in [n]})$:
 $miSK \leftarrow miKeyGen(miMSK, \{(\sum_{j \in [n]} c_{j, i} u_j \tilde{u}_i, \sum_{j \in [n]} c_{i, j} v_i \tilde{v}_j)\}_{i \in [n]})$
 $qSK := (miSK, \{-c_{i, j} w_i \tilde{w}_j\}_{i, j \in [n]})$.

$$\begin{aligned} & \text{qDec}(\text{qCT}_1, \dots, \text{qCT}_n, \text{qSK}): \\ & - \sum_{i,j \in [n]} c_{i,j} w_i \tilde{w}_j \text{iDec}(\text{iCT}'_i, \text{iSK}'_j) + \sum_{i,j \in [n]} c_{i,j} \text{pDec}(\text{pCT}_i, \text{pSK}_j) \\ & - \text{miDec}(\text{miCT}_1, \dots, \text{miCT}_n, \text{miSK}) = [\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle]_T \end{aligned}$$

However, while the above candidate satisfies functionality and resists the aforementioned attacks, we are still far from a proof of security. For instance, one hurdle is that we must argue that $\{w_i \tilde{w}_j\}_{i,j \in [n]}$ is pseudorandom, which is not true because qSK contains these elements not as exponents of group elements but as elements in \mathbb{Z}_p . Moreover, since we have already “used up” our pairing, we cannot move these to the exponent as in [28]. Another hurdle is that the underlying IPFE schemes satisfy only indistinguishability based security rather than simulation based security. To arrive at a security proof, we must address several such challenges, which we describe next.

Overview of Proof of Security. For ease of exposition, we outline our ideas for the warm-up case of two input quadratic MIFE described in Sect. 5. The general case is handled in Sect. 6.

First, we briefly recall the definition for indistinguishability based security of secret-key MIFE. Intuitively, security requires that all PPT adversaries cannot guess a randomly chosen bit β with meaningful probability in the following game: the adversary first outputs a set of challenge messages $\{i, x_i^{j,0}, x_i^{j,1}\}_{i \in [n], j \in [q_{\text{CT}}]}$ and obtains ciphertexts for $\{i, x_i^{j,\beta}\}$. After that, the adversary can query a key generation oracle on any functions f such that for all $(j_1, \dots, j_n) \in [q_{\text{CT}}]^n$, it holds that $f(x_1^{j_1,0}, \dots, x_n^{j_n,0}) = f(x_1^{j_1,1}, \dots, x_n^{j_n,1})$. The goal of the security proof is to show that ciphertexts for $\{i, x_i^{j,0}\}$ and $\{i, x_i^{j,1}\}$ are indistinguishable.

The first challenge in the security proof is how to design a series of hybrids between the real games G^β for $\beta = 0$ and $\beta = 1$. A naive strategy is to change each ciphertext from $\beta = 0$ to $\beta = 1$ one by one, that is, in hybrid H_ι^η for $\iota \in [2], \eta \in [q_{\text{CT}}]$, the adversary is given the ciphertext for $x_i^{j,1}$ if $(i, j) \leq (\iota, \eta)$ and that for $x_i^{j,0}$ otherwise, where $(i, j) \leq (\iota, \eta) \Leftrightarrow (i-1)q_{\text{CT}} + j \leq (\iota-1)q_{\text{CT}} + \eta$. Then, we may hope to prove that $G^0 \approx_c H_1^0 \approx_c \dots \approx_c H_1^{q_{\text{CT}}} \approx_c H_2^0 \approx_c \dots \approx_c H_2^{q_{\text{CT}}} \approx_c G^1$. However, it quickly becomes evident that this strategy does not work. This is since the queried function f does not necessarily satisfy $f(x_1^{1,0}, x_2^{j_2,0}) = f(x_1^{1,1}, x_2^{j_2,0})$, and thus the adversary can trivially distinguish G^0 from H_1^1 . Even worse, when we change some input from $\beta = 0$ to $\beta = 1$, the change affects the quadratic terms that contain an input from another slot such as $x_1^{1,1} x_2^{j_2,0}$. This correlation does not appear in IP-MIFE and makes the proof much more complex.

We address this issue as follows. Recall that our quadratic MIFE decryption first generates modified ACFGU ciphertexts $\{\text{aCT}_{i,\ell}\}_{i,\ell \in [2]}$ and a secret key element aSK where

$$\begin{aligned} \text{aCT}_{i,\ell} &= \text{pDec}(\text{pCT}_i, \text{pSK}_\ell) = [x_i x_\ell + s_i \tilde{s}_\ell w_i \tilde{w}_\ell + r_\ell u_i \tilde{u}_\ell + t_i v_i \tilde{v}_\ell]_T \\ \text{aSK} &= \text{miDec}(\text{miCT}_1, \text{miCT}_2, \text{miSK}) = \left[- \sum_{i,\ell \in [2]} c_{i,\ell} (r_\ell u_i \tilde{u}_\ell + t_i v_i \tilde{v}_\ell)\right]_T. \end{aligned}$$

Our first idea is to define H_l^η so that $\text{qDec}(\text{qCT}_1^{j_1}, \text{qCT}_2^{j_2}, \text{qSK})$ in H_l^η yields $(\{\text{aCT}_{i,\ell}^{j_i,j_\ell}\}_{i,\ell \in [2]}, \text{aSK}^{j_1,j_2})$ where

$$\begin{aligned} \text{aCT}_{i,\ell}^{j_i,j_\ell} &= \begin{cases} [x_i^1 x_\ell^1 + s_i \tilde{s}_\ell w_i \tilde{w}_\ell + r_\ell u_i \tilde{u}_\ell + t_i v_i \tilde{v}_\ell]_T & (\ell, j_\ell) \leq (\iota, \eta) \\ [x_i^0 x_\ell^0 + s_i \tilde{s}_\ell w_i \tilde{w}_\ell + r_\ell u_i \tilde{u}_\ell + t_i v_i \tilde{v}_\ell]_T & (\ell, j_\ell) > (\iota, \eta) \end{cases} \\ \text{aSK}^{j_1,j_2} &= [-\sum_{i,\ell \in [2]} c_{i,\ell} (r_\ell u_i \tilde{u}_\ell + t_i v_i \tilde{v}_\ell) - \sum_{\substack{i \in [2] \\ \ell \in \{k \in [2] \mid (k, j_k) \leq (\iota, \eta)\}}} c_{i,\ell} (x_i^1 x_\ell^1 - x_i^0 x_\ell^0)]_T. \end{aligned}$$

Note that variables x, s, \tilde{s}, r, t are also indexed by j_1, j_2 , but we often omit j_1, j_2 for conciseness if it is clear in context. Observe that, in hybrid H_l^η , $\sum_{i,\ell \in [2]} c_{i,\ell} \text{aCT}_{i,\ell}^{j_i,j_\ell} + \text{aSK}^{j_1,j_2} = \sum_{i,\ell \in [2]} c_{i,\ell} [x_i^0 x_\ell^0 + s_i \tilde{s}_\ell w_i \tilde{w}_\ell]_T$ for all $(\iota, \eta, j_1, j_2) \in [2] \times [q_{\text{CT}}]^3$. Therefore, the adversary always obtains $f(x_1^0, x_2^0)$ by decryption in all hybrids and cannot trivially distinguish them. Since the second term of aSK^{j_1,j_2} , $\sum_{i,\ell \in [2]} c_{i,\ell} (x_i^1 x_\ell^1 - x_i^0 x_\ell^0) = 0$ due to the query condition, H_2^{qCT} almost can be seen as G^1 . Thanks to the function-hiding property of pFE and miFE, information encoded in ciphertexts and secret keys is not revealed other than $\text{aCT}_{i,\ell}, \text{aSK}$.

Next we must define encoded vectors in ciphertexts and secret keys in pFE and miFE in each hybrid so that they are indistinguishable in the hybrid sequence. First, let us consider vectors encoded in pFE that yield $\text{aCT}_{i,\ell}$. In G^0 , recall that $\mathbf{b}_i = (x_i^0, s_i w_i, u_i, t_i v_i)$ and $\tilde{\mathbf{b}}_i = (x_i^0, \tilde{s}_i \tilde{w}_i, r_i \tilde{u}_i, \tilde{v}_i)$ are encoded in pCT_i and pSK_i , respectively. To make $\langle \mathbf{b}_i^{j_i}, \tilde{\mathbf{b}}_i^{j_\ell} \rangle_T = \text{aCT}_{i,\ell}^{j_i,j_\ell}$ in all hybrids, we introduce a free space, used for only the security proof, and define $\mathbf{b}_i^{j_i}, \tilde{\mathbf{b}}_i^{j_i}$ in H_l^η as follows:

$$\mathbf{b}_i^{j_i} = (x_i^0, \underline{x_i^1}, s_i w_i, u_i, t_i v_i), \quad \tilde{\mathbf{b}}_i^{j_i} = \begin{cases} (0, x_i^1, \tilde{s}_i \tilde{w}_i, r_i \tilde{u}_i, \tilde{v}_i) & (i, j_i) \leq (\iota, \eta) \\ (x_i^0, \underline{0}, \tilde{s}_i \tilde{w}_i, r_i \tilde{u}_i, \tilde{v}_i) & (i, j_i) > (\iota, \eta) \end{cases}.$$

Then, we need to prove that $\{\mathbf{b}_i^{j_i}, \tilde{\mathbf{b}}_i^{j_i}\}_{i \in [2], j_i \in [q_{\text{CT}}]}$ in $H_l^{\eta-1}$ and that in H_l^η are indistinguishable. Initially, it appears that we can prove it similarly to Lin's technique [28], that is, we introduce a more free space and consider an intermediate hybrid in which we define

$$\begin{aligned} \mathbf{b}_i^{j_i} &= (x_i^{j_i,0}, x_i^{j_i,1}, s_i w_i, u_i, t_i v_i, \underline{x_i^{j_i,0} x_l^{\eta,0} + s_i \tilde{s}_l w_l \tilde{w}_l + r_l u_l \tilde{u}_l + t_i v_i \tilde{v}_l}) \quad (1.1) \\ \tilde{\mathbf{b}}_i^{j_i} &= \begin{cases} (0, x_i^{j_i,1}, \tilde{s}_i \tilde{w}_i, r_i \tilde{u}_i, \tilde{v}_i, \underline{0}) & (i, j_i) < (\iota, \eta) \\ (0, 0, 0, 0, 0, 1) & (i, j_i) = (\iota, \eta) \\ (x_i^{j_i,0}, 0, \tilde{s}_i \tilde{w}_i, r_i \tilde{u}_i, \tilde{v}_i, \underline{0}) & (i, j_i) > (\iota, \eta) \end{cases} \end{aligned}$$

Now, we may hope to change $x_i^{j_i,0} x_l^{\eta,0}$ in the last entry of $\mathbf{b}_i^{j_i}$ to $x_i^{j_i,1} x_l^{\eta,1}$ by the indistinguishability-based security of the (modified) ACFGU IP-MIFE scheme.

However, we get stuck here; the relation between $\{x_i^{j_i,0} x_l^{\eta,0}\}_{i \in [2], j_i \in [q_{\text{CT}}]}$ and $\{x_i^{j_i,1} x_l^{\eta,1}\}_{i \in [2], j_i \in [q_{\text{CT}}]}$ implied by the query condition $f(x_1^{j_1,0}, x_2^{j_2,0}) = f(x_1^{j_1,1}, x_2^{j_2,1})$ is unclear. This is because, in the reduction to ACFGU IP-MIFE, the simulator is expected to simulate pCT for $\mathbf{b}_i^{j_i}$ and qSK for quadratic function f using ACFGU ciphertexts for $\{x_i^{j_i,\beta} x_l^{\eta,\beta}\}_{i \in [2], j_i \in [q_{\text{CT}}]}$ and secret

keys for linear functions f_ι , respectively, such that $f_\iota(x_1^{j_1,0}x_\iota^{\eta,0}, x_2^{j_2,0}x_\iota^{\eta,0}) = f_\iota(x_1^{j_1,1}x_\iota^{\eta,1}, x_2^{j_2,1}x_\iota^{\eta,1})$. Note that f_ι comprises coefficients of f that are related to the ι -th input. Unfortunately, we cannot derive the above relation on f_ι from the query condition. The critical observation we make here is that we have an alternative equality on f_ι that are implied by the condition: for all $(j_1, j_2, \eta) \in [q_{CT}]^3$, we have

$$f_1(x_1^{\eta,0}x_1^{\eta,0} - x_1^{1,0}x_1^{1,0}, x_2^{j_2,0}x_1^{\eta,0} - x_2^{j_2,0}x_1^{1,0}) = f_1(x_1^{\eta,1}x_1^{\eta,1} - x_1^{1,1}x_1^{1,1}, x_2^{j_2,1}x_1^{\eta,1} - x_2^{j_2,1}x_1^{1,1}) \quad (1.2)$$

$$f_2(x_1^{j_1,0}x_2^{\eta,0} - x_1^{j_1,0}x_2^{1,0}, x_2^{\eta,0}x_2^{\eta,0} - x_2^{1,0}x_2^{1,0}) = f_2(x_1^{j_1,1}x_2^{\eta,1} - x_1^{j_1,1}x_2^{1,1}, x_2^{\eta,1}x_2^{\eta,1} - x_2^{1,1}x_2^{1,1}). \quad (1.3)$$

Equation (1.2) and (1.3) can be obtained by Eq. (1.4)–Eq. (1.5) where

$$f(x_1^{\eta,0}, x_2^{j_2,0}) = f(x_1^{\eta,1}, x_2^{j_2,1}) \quad f(x_1^{j_1,0}, x_2^{\eta,0}) = f(x_1^{j_1,1}, x_2^{\eta,1}) \quad (1.4)$$

$$f(x_1^{1,0}, x_2^{j_2,0}) = f(x_1^{1,1}, x_2^{j_2,1}) \quad f(x_1^{j_1,0}, x_2^{1,0}) = f(x_1^{j_1,1}, x_2^{1,1}). \quad (1.5)$$

The last challenge is to somehow change $x_i^{j_i,0}x_\iota^{\eta,0}$ in the last entry of Eq. (1.1) in to $x_i^{j_i,1}x_\iota^{\eta,1}$ leveraging Eq. (1.2) or Eq. (1.3). We first observe that

$$\begin{aligned} x_i^{j_i,0}x_\iota^{\eta,0} + s_i^{j_i}\widehat{s}_\iota^i w_i \widehat{w}_\iota + r_\iota^{j_i} u_i \widehat{u}_\iota + t_i^{j_i} v_i \widehat{v}_\iota &\approx_c x_i^{j_i,0}x_\iota^{\eta,0} + \widehat{s}_{i,\iota}^{j_i} \widehat{w}_{i,\iota} + \widehat{u}_i + \widehat{v}_i^{j_i} \\ &= \underbrace{x_i^{j_i,0}x_\iota^{\eta,0} - x_i^{j_i,0}x_\iota^{1,0} + \widehat{s}_{i,\iota}^{j_i} \widehat{w}_{i,\iota} + \widehat{u}_i + \widehat{v}_i^{j_i}}_{\text{ACFGU ciphertext}} \end{aligned}$$

where $\widehat{s}_{i,\iota}^{j_i}, \widehat{w}_{i,\iota}, \widehat{u}_i, \widehat{v}_i^{j_i}, \widehat{v}_i^{j_i}$ are fresh random elements. The computational indistinguishability is implied by the SXDH assumption, and the equality follows by implicitly defining $\widehat{v}_i^{j_i} = \widehat{v}_i^{j_i} - x_i^{j_i,0}x_\iota^{1,0}$. We can see that the last part of the above equation is exactly the ACFGU ciphertext of $x_i^{j_i,0}x_\iota^{\eta,0} - x_i^{j_i,0}x_\iota^{1,0}$ plus $\widehat{v}_i^{j_i}$. At this point, we can use the security of the ACFGU IP-MIFE scheme to change $x_i^{j_i,0}x_\iota^{\eta,0} - x_i^{j_i,0}x_\iota^{1,0}$ to $x_i^{j_i,1}x_\iota^{\eta,1} - x_i^{j_i,1}x_\iota^{1,1}$. This is because they satisfy Eq. (1.2) or Eq. (1.3), and thus the reduction can follow the query condition of IP-MIFE. Perceptive readers may notice that if $i = \iota$, then $x_i^{j_i,0}x_\iota^{\eta,0} - x_i^{j_i,0}x_\iota^{1,0} = x_i^{j_i,1}x_\iota^{\eta,1} - x_i^{j_i,1}x_\iota^{1,1}$ holds only when $j_i = \eta$. This is not a problem since we can deal with the terms for $i = \iota, j_i \neq \eta$ leveraging the security of predicated IPFE.

Next we give some intuition for how to define vectors in miFE. Similarly to $\mathbf{b}_i^{j_i}, \widetilde{\mathbf{b}}_i^{j_i}$, we want to define $\mathbf{f}_i^{j_i}, \widetilde{\mathbf{f}}_i$ in \mathbf{H}_ℓ^η , which are encoded in miFE and yield aSK, but this approach quickly runs into cumbersome issues. The first problem is that the second term of $\text{aSK}^{j_1, j_2}, \text{aSK}^{j_1, j_2}[2] = \sum c_{i,\ell}(x_i^{j_1,1}x_\ell^{j_2,1} - x_i^{j_1,0}x_\ell^{j_2,0})$, in the current definition depends on both $x_1^{j_1}$ and $x_2^{j_2}$. Thus, we must somehow encode $x_1^{j_1}$ and $x_2^{j_2}$ in $\text{miCT}_1^{j_1}$ and $\text{miCT}_2^{j_2}$, respectively. However, we cannot generate the term $x_1^{j_1}x_2^{j_2}$ via miFE, which can only compute linear functions! A naive idea may be to program all quadratic terms into additional free spaces in miCT. It immediately ends in failure; we cannot program q_{CT}^2 values into $O(q_{CT})$ spaces.

Our solution is to use Eq. (1.2) and Eq. (1.3) to compress the q_{CT}^2 values into q_{CT} values. For instance, Eq. (1.2) implies

$$f_1(x_1^{j_1,1}x_1^{j_1,1} - x_1^{j_1,0}x_1^{j_1,0}, x_2^{j_2,1}x_1^{j_1,1} - x_2^{j_2,0}x_1^{j_1,0}) = f_1(x_1^{1,1}x_1^{1,1} - x_1^{1,0}x_1^{1,0}, x_2^{j_2,1}x_1^{1,1} - x_2^{j_2,0}x_1^{1,0})$$

since f_1 is a linear function (we change η to j_1). This means that $\sum_{\ell=1} c_{i,\ell}(x_i^{j_i,1} x_\ell^{j_\ell,1} - x_i^{j_i,0} x_\ell^{j_\ell,0}) = \sum_{\ell=1} c_{i,\ell}(x_i^{j_i,1} x_\ell^{1,1} - x_i^{j_i,0} x_\ell^{1,0})$ for all j_i . Similarly, we can handle the case for $\ell = 2$. Thus, we can program $\text{aSK}^{j_1, j_2}[2]$ in $\text{miCT}_1^{j_1}$ and $\text{miCT}_2^{j_2}$ as:

$$\mathbf{f}_i^{j_i} = \begin{cases} (r_i, t_i, \frac{x_i^{j_i,1} x_1^{1,1} - x_i^{j_i,0} x_1^{1,0}}{x_i^{j_i,1} x_1^{1,1} - x_i^{j_i,0} x_1^{1,0}}, 0) & \iota = 1 \\ (r_i, t_i, \frac{x_i^{j_i,1} x_1^{1,1} - x_i^{j_i,0} x_1^{1,0}}{x_i^{j_i,1} x_1^{1,1} - x_i^{j_i,0} x_1^{1,0}}, \frac{x_i^{j_i,1} x_2^{1,1} - x_i^{j_i,0} x_2^{1,0}}{x_i^{j_i,1} x_1^{1,1} - x_i^{j_i,0} x_1^{1,0}}) & \iota = 2 \end{cases}$$

$$\tilde{\mathbf{f}}_i = (\sum_{\ell \in [2]} c_{\ell,i} u_\ell \tilde{u}_i, \sum_{\ell \in [2]} c_{i,\ell} v_i \tilde{v}_\ell, \underline{c_{i,1}, c_{i,2}}).$$

The second problem is the fact that

$$\overline{\text{aSK}^{j_1, j_2}[2]} = \langle \mathbf{f}_i^{j_i}, \tilde{\mathbf{f}}_i \rangle - \sum_{i, \ell \in [2]} c_{i,\ell} (r_\ell u_i \tilde{u}_\ell + t_i v_i \tilde{v}_\ell) = \sum_{i \in [2], \ell \in [l]} c_{i,\ell} (x_i^1 x_\ell^1 - x_i^0 x_\ell^0)$$

in the current definition of $\mathbf{f}_i^{j_i}, \tilde{\mathbf{f}}_i$, while $\text{aSK}^{j_1, j_2}[2]$ should be

$$\text{aSK}^{j_1, j_2}[2] = \sum_{\substack{i \in [2] \\ \ell \in \{k \in [2] \mid (k, j_k) \leq (\iota, \eta)\}}} c_{i,\ell} (x_i^1 x_\ell^1 - x_i^0 x_\ell^0).$$

We adjust them by modifying aCT as $\overline{\text{aCT}_{i,\ell}^{j_i, j_\ell}} = \text{aCT}_{i,\ell}^{j_i, j_\ell} + Q(\mathbf{x})$ so that $\sum_{i, \ell \in [2]} c_{i,\ell} \overline{\text{aCT}_{i,\ell}^{j_i, j_\ell}} + \text{aSK}^{j_1, j_2} = \sum_{i, \ell \in [2]} c_{i,\ell} [x_i^0 x_\ell^0 + s_i \tilde{s}_\ell w_i \tilde{w}_\ell]_T$ holds, where Q is a quadratic polynomial over variables $\mathbf{x} = \{x_i^{j_i, \beta}\}_{i \in [2], j_i \in [q_{\text{CT}}], \beta \in \{0,1\}}$. The additional term $Q(\mathbf{x})$ in $\overline{\text{aCT}_{i,\ell}^{j_i, j_\ell}}$ can be programmed into pCT and pSK by introducing more additional space. Please see Sect. 5 for a detailed argument.

2 Preliminaries

In this section, we define some notation and preliminaries that we require. For vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$, $(\mathbf{v}_1, \dots, \mathbf{v}_n)$ denotes the vector concatenation as row vectors *regardless of* whether each \mathbf{v}_i is a row or column vector. We use \otimes for the Kronecker product. We denote an n -dimensional unit vector $(0^{i-1}, 1, 0^{n-1})$ by $\mathbf{e}_{i/n}$. We use standard cryptographic bilinear groups where the matrix decisional Diffie-Hellman assumption (MDDH) holds [18].

2.1 Multi-input Functional Encryption

Definition 2.1 (Multi-Input Functional Encryption). Let \mathcal{F} be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X}_1 \times \dots \times \mathcal{X}_n \rightarrow \mathcal{Z}$. An MIFE scheme for \mathcal{F} , MIFE, consists of four algorithms.

Setup(1^λ): It takes a security parameter 1^λ and outputs a public parameter PP and a master secret key MSK . The other algorithms implicitly take PP .

$\text{Enc}(\text{MSK}, i, x_i)$: It takes MSK, an index $i \in [n]$, and $x_i \in \mathcal{X}_i$ and outputs a ciphertext CT_i .

$\text{KeyGen}(\text{MSK}, f)$: It takes MSK, and $f \in \mathcal{F}$, and outputs a secret key SK.

$\text{Dec}(\text{CT}_1, \dots, \text{CT}_n, \text{SK})$: It takes $\text{CT}_1, \dots, \text{CT}_n$ and SK, and outputs a decryption value $d \in \mathbb{Z}$ or a symbol \perp .

When $n = 1$, we call it just a functional encryption (FE) scheme and omit the second argument of Enc .

Correctness. MIFE is *correct* if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $(x_1, \dots, x_n) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, $f \in \mathcal{F}$, we have

$$\Pr \left[d = f(x_1, \dots, x_n) \mid \begin{array}{l} \text{PP, MSK} \leftarrow \text{Setup}(1^\lambda) \\ \text{CT}_i \leftarrow \text{Enc}(\text{MSK}, i, x_i) \\ \text{SK} \leftarrow \text{KeyGen}(\text{MSK}, f) \\ d := \text{Dec}(\text{CT}_1, \dots, \text{CT}_n, \text{SK}) \end{array} \right] = 1.$$

Selective Security. We define two indistinguishability-based security definitions for MIFE, namely, message-hiding and function-hiding. For a stateful PPT adversary \mathcal{A} and $\lambda \in \mathbb{N}$, let

$$\mathbf{P}_{\mathcal{A}, \text{mh}}^{\text{MIFE}, \beta}(\lambda) := \Pr \left[\beta' = 1 \mid \begin{array}{l} \{i, x_i^{j_i, 0}, x_i^{j_i, 1}\}_{i \in [n], j_i \in [q_{\text{CT}, i}]} \leftarrow \mathcal{A}(1^\lambda) \\ \text{PP, MSK} \leftarrow \text{Setup}(1^\lambda), \\ \text{CT}_i^{j_i} \leftarrow \text{Enc}(\text{MSK}, i, x_i^{j_i, \beta}) \\ \beta' \leftarrow \mathcal{A}^{\text{KeyGen}(\text{MSK}, \cdot)}(\text{PP}, \{\text{CT}_i^{j_i}\}_{i \in [n], j_i \in [q_{\text{CT}, i}]}) \end{array} \right].$$

Let q_{SK} be a number of queries to KeyGen . We say \mathcal{A} is *admissible* if, in case of $q_{\text{CT}, 1}, \dots, q_{\text{CT}, n}, q_{\text{SK}} \geq 1$, \mathcal{A} 's queries satisfy $f^\ell(x_1^{j_1, 0}, \dots, x_n^{j_n, 0}) = f^\ell(x_1^{j_1, 1}, \dots, x_n^{j_n, 1})$ for all $(j_1, \dots, j_n) \in [q_{\text{CT}, 1}] \times \dots \times [q_{\text{CT}, n}]$ and $\ell \in [q_{\text{SK}}]$. MIFE is *message-hiding* if, for all admissible PPT adversaries \mathcal{A} , the following advantage of \mathcal{A} is negligible in λ : $\text{Adv}_{\mathcal{A}, \text{mh}}^{\text{MIFE}}(\lambda) := |\mathbf{P}_{\mathcal{A}, \text{mh}}^{\text{MIFE}, 0}(\lambda) - \mathbf{P}_{\mathcal{A}, \text{mh}}^{\text{MIFE}, 1}(\lambda)|$.

Next, we define a function-hiding property. Let $\mathbf{P}_{\mathcal{A}, \text{fh}}^{\text{MIFE}, \beta}(\lambda)$ be defined the same as $\mathbf{P}_{\mathcal{A}, \text{mh}}^{\text{MIFE}, \beta}(\lambda)$ except that \mathcal{A} 's oracle is $\mathcal{O}_{\text{SK}}(\beta, \cdot)$ instead of KeyGen , where $\mathcal{O}_{\text{SK}}(\beta, \cdot)$ takes (f^0, f^1) and outputs $\text{KeyGen}(\text{MSK}, f^\beta)$. This time, \mathcal{A} is *admissible* if, in case of $q_{\text{CT}, 1}, \dots, q_{\text{CT}, n}, q_{\text{SK}} \geq 1$, \mathcal{A} 's queries satisfy $f^{\ell, 0}(x_1^{j_1, 0}, \dots, x_n^{j_n, 0}) = f^{\ell, 1}(x_1^{j_1, 1}, \dots, x_n^{j_n, 1})$ for all $(j_1, \dots, j_n) \in [q_{\text{CT}, 1}] \times \dots \times [q_{\text{CT}, n}]$ and $\ell \in [q_{\text{SK}}]$. Then, MIFE is *function-hiding* if, for all admissible PPT adversaries \mathcal{A} , the following advantage of \mathcal{A} is negligible in λ : $\text{Adv}_{\mathcal{A}, \text{fh}}^{\text{MIFE}}(\lambda) := |\mathbf{P}_{\mathcal{A}, \text{fh}}^{\text{MIFE}, 0}(\lambda) - \mathbf{P}_{\mathcal{A}, \text{fh}}^{\text{MIFE}, 1}(\lambda)|$.

Remark 2.1. In this paper, we assume that $q_{\text{CT}, i} \geq 1$ for all $i \in [n]$ and that $q_{\text{CT}, 1} = \dots = q_{\text{CT}, n} (= q_{\text{CT}})$. This is w.l.o.g as discussed in [6, 17].

We next define quadratic functions.

Definition 2.2 (Bounded-Norm Multi-Input Quadratic functions over \mathbb{Z}). A function family $\mathcal{F}_{m, n, X, C}^{\text{MQF}}$ for bounded-norm multi-input quadratic functions consist of functions $f : (\mathcal{X}^m)^n \rightarrow \mathbb{Z}$ where $\mathcal{X} = \{i \mid i \in \mathbb{Z}, |i| \leq X\}$. Each $f \in \mathcal{F}_{m, n, X, C}^{\text{MQF}}$ is specified by $\mathbf{c} = \{c_{\mu, \nu}\}_{\mu, \nu \in [mn]} \in \mathbb{Z}^{(mn)^2}$ s.t. $\|\mathbf{c}\|_\infty \leq C$ and $c_{\mu, \nu} = 0$ if $\mu > \nu$. Let x_μ be the μ -th element of $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n) \in (\mathcal{X}^m)^n$. Then, f specified by \mathbf{c} is defined as $f(\mathbf{x}_1, \dots, \mathbf{x}_n) := \sum_{\mu, \nu \in [mn]} c_{\mu, \nu} x_\mu x_\nu$.

3 Predicated Inner Product Functional Encryption

We define and construct predicated inner product functional encryption.

Definition 3.1 (Inner Products over Bilinear Groups). Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. A function family $\mathcal{F}_{m,\mathbb{G}}^{\text{IP}}$ for inner products over bilinear groups consists of functions $f : G_1^m \rightarrow G_T$. Each $f \in \mathcal{F}_{m,\mathbb{G}}^{\text{IP}}$ is specified by $[\mathbf{y}]_2$ where $\mathbf{y} \in \mathbb{Z}_p^m$ and defined as $f([\mathbf{x}]_1) := \langle \mathbf{x}, \mathbf{y} \rangle_T$.

Definition 3.2 (Predicated Inner Products over Bilinear Groups). A function family $\mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$ for predicated inner products over bilinear groups consists of functions $f : \mathbb{Z}_p^d \times G_1^m \rightarrow G_T \cup \{\perp\}$. Each $f \in \mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$ is specified by $\mathbf{y}_1 \in \mathbb{Z}_p^d$ and $[\mathbf{y}_2]_2$ where $\mathbf{y}_2 \in \mathbb{Z}_p^m$ and defined as $f(\mathbf{x}_1, [\mathbf{x}_2]_1) := \begin{cases} \langle \mathbf{x}_2, \mathbf{y}_2 \rangle_T & \text{if } \langle \mathbf{x}_1, \mathbf{y}_1 \rangle = 0 \\ \perp & \text{if } \langle \mathbf{x}_1, \mathbf{y}_1 \rangle \neq 0 \end{cases}$.

We refer to FE for $\mathcal{F}_{m,\mathbb{G}}^{\text{IP}}$ and $\mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$ as IPFE and predicated IPFE, respectively. We define partially function-hiding security of FE for $\mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$. Partially function-hiding security guarantees that secret keys hide \mathbf{y}_2 (but do not \mathbf{y}_1).

Partially Function-Hiding Security. Let pFE = (pSetup, pEnc, pKeyGen, pDec) be a FE scheme for $\mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$. For a stateful PPT adversary \mathcal{A} and $\lambda \in \mathbb{N}$, let

$$\text{P}_{\mathcal{A},\text{pFH}}^{\text{pFE},\beta}(\lambda) := \Pr \left[\beta' = 1 \left[\begin{array}{l} \{\mathbf{x}_1^j, [\mathbf{x}_2^j]_1, [\mathbf{x}_2^j]_1\}_{j \in [q_{\text{CT}}]} \leftarrow \mathcal{A}(1^\lambda) \\ \text{pPP}, \text{pMSK} \leftarrow \text{pSetup}(1^\lambda), \\ \text{pCT}^j \leftarrow \text{pEnc}(\text{pMSK}, (\mathbf{x}_1^j, [\mathbf{x}_2^j]_1)) \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SK}}(\beta, \cdot)}(\text{pPP}, \{\text{pCT}^j\}_{j \in [q_{\text{CT}}]}) \end{array} \right] \right]$$

where \mathcal{O}_{SK} takes $(\mathbf{y}_1, [\mathbf{y}_2^0]_2, [\mathbf{y}_2^1]_2)$ and outputs $\text{pKeyGen}(\text{MSK}, (\mathbf{y}_1, [\mathbf{y}_2^\beta]_2))$. Let q_{SK} be a number of queries to \mathcal{O}_{SK} . We say \mathcal{A} is *admissible* if \mathcal{A} 's queries satisfy $\langle \mathbf{x}_2^{j,0}, \mathbf{y}_2^{\ell,0} \rangle = \langle \mathbf{x}_2^{j,1}, \mathbf{y}_2^{\ell,1} \rangle$ when $\langle \mathbf{x}_1^j, \mathbf{y}_1^\ell \rangle = 0$ for all $j \in [q_{\text{CT}}]$ and $\ell \in [q_{\text{SK}}]$. pFE is *partially function-hiding* if, for all admissible PPT adversaries \mathcal{A} , the following advantage of \mathcal{A} is negligible in λ : $\text{Adv}_{\mathcal{A},\text{pFH}}^{\text{pFE}}(\lambda) := |\text{P}_{\mathcal{A},\text{pFH}}^{\text{pFE},0}(\lambda) - \text{P}_{\mathcal{A},\text{pFH}}^{\text{pFE},1}(\lambda)|$.

3.1 Predicated IPFE from IPFE

We construct a partially function-hiding FE scheme for $\mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$ from a function-hiding FE scheme for $\mathcal{F}_{kd+2m+1,\mathbb{G}}^{\text{IP}}$ generically. Note that k is a parameter for the MDDH assumption. A function-hiding FE scheme for $\mathcal{F}_{m,\mathbb{G}}^{\text{IP}}$ based on MDDH is implied by the function-hiding IPFE scheme described in [30, Appx. A]⁴. Let iFE = (iSetup, iEnc, iKeyGen, iDec) be a function-hiding FE scheme for $\mathcal{F}_{kd+2m+1,\mathbb{G}}^{\text{IP}}$. Then, our partially function-hiding FE scheme pFE = (pSetup, pEnc, pKeyGen, pDec) for $\mathcal{F}_{d,m,\mathbb{G}}^{\text{PIP}}$ is constructed as shown in Fig. 1.

⁴ In more detail, this follows since the scheme remains correct and secure even if input vectors for Enc and KeyGen consist of group elements, and Dec first obtains decryption values on the exponent of a target-group generator and then computes its discrete log.

$\text{pSetup}(1^\lambda) \rightarrow \text{pPP}, \text{pMSK}$ $(\text{pPP}, \text{pMSK}) := (\text{iPP}, \text{iMSK}) \leftarrow \text{iSetup}(1^\lambda)$
$\text{pEnc}(\text{MSK}, (\mathbf{x}_1, [\mathbf{x}_2]_1)) \rightarrow \text{pCT}$ $\mathbf{z} \leftarrow \mathbb{Z}_p^k, \mathbf{y} := (\mathbf{z} \otimes \mathbf{x}_1, \mathbf{x}_2, 0^m, 0) \in \mathbb{Z}_p^{kd+2m+1}, \text{iCT} \leftarrow \text{iEnc}(\text{iMSK}, [\mathbf{x}]_1), \text{pCT} := (\mathbf{x}_1, \text{iCT})$
$\text{pKeyGen}(\text{pMSK}, (\mathbf{y}_1, [\mathbf{y}_2]_2)) \rightarrow \text{pSK}$ $\mathbf{a} \leftarrow \mathbb{Z}_p^k, \mathbf{y} := (\mathbf{a} \otimes \mathbf{y}_1, \mathbf{y}_2, 0^m, 0) \in \mathbb{Z}_p^{kd+2m+1}, \text{iSK} \leftarrow \text{iKeyGen}(\text{iMSK}, [\mathbf{y}]_2), \text{pSK} := (\mathbf{y}_1, \text{iSK})$
$\text{pDec}(\text{pCTpSK}) \rightarrow z$ <p>If $\langle \mathbf{x}_1, \mathbf{y}_1 \rangle \neq 0$, outputs $z = \perp$. Otherwise, outputs $z = \text{iDec}(\text{iCT}, \text{iSK})$.</p>

Fig. 1. Our predicated IPFE scheme.

Correctness. Since $\langle \mathbf{z} \otimes \mathbf{x}_1, \mathbf{a} \otimes \mathbf{y}_1 \rangle = \langle \mathbf{z}, \mathbf{a} \rangle \cdot \langle \mathbf{x}_1, \mathbf{y}_1 \rangle$, $\text{iDec}(\text{iCT}, \text{iSK})$ outputs $[\langle \mathbf{x}, \mathbf{y} \rangle]_T = [\langle v_2, \mathbf{y}_2 \rangle]_T$ if $\langle \mathbf{x}_1, \mathbf{y}_1 \rangle = 0$. This follows from the correctness of iFE.

For security, we have the following theorem.

Theorem 3.1. *If iFE is function-hiding, and the MDDH assumption holds in \mathbb{G} , then pFE is partially function-hiding. More precisely, for all PPT adversaries \mathcal{A} , there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2$ such that*

$$\text{Adv}_{\mathcal{A}, \text{pFH}}^{\text{pFE}}(\lambda) \leq q_{\text{CT}}(3\text{Adv}_{\mathcal{B}_1, \text{FH}}^{\text{iFE}}(\lambda) + 2\text{Adv}_{\mathcal{B}_2}^{\mathcal{D}_k\text{-MDDH}}(\lambda)).$$

Due to space constraints, the proof is provided in the full version.

4 Mixed-Group Multi-input IPFE

In this section, we define and construct our mixed-group multi-input inner product functional encryption (mixed-group IP-MIFE).

Definition 4.1 (Multi-Input Inner Products over Bilinear Groups). Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. A function family $\mathcal{F}_{m,n,\mathbb{G}}^{\text{MIP}}$ for multi-input inner products over bilinear groups consists of functions $f : (G_1^m)^n \rightarrow G_T$. Each $f \in \mathcal{F}_{m,n,\mathbb{G}}^{\text{MIP}}$ is specified by $[\mathbf{y}_1]_2, \dots, [\mathbf{y}_n]_2$ where $\mathbf{y}_i \in \mathbb{Z}_p^m$ and defined as $f([\mathbf{x}]_1, \dots, [\mathbf{x}]_n) := [\sum_{i \in [n]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle]_T$.

Definition 4.2 (Multi-Input Mixed-Group Inner Products over Bilinear Groups). Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. A function family $\mathcal{F}_{m_1, m_2, n, \mathbb{G}}^{\text{MGIP}}$ for multi-input mixed-group inner products over bilinear groups consists of functions $f : (G_1^{m_1} \times G_2^{m_2})^n \rightarrow G_T$. Each $f \in \mathcal{F}_{m_1, m_2, n, \mathbb{G}}^{\text{MGIP}}$ is specified by $([\mathbf{y}_{1,1}]_2, [\mathbf{y}_{1,2}]_1, \dots, [\mathbf{y}_{n,1}]_2, [\mathbf{y}_{n,2}]_1)$ where $\mathbf{y}_{i,1} \in \mathbb{Z}_p^{m_1}$ and $\mathbf{y}_{i,2} \in \mathbb{Z}_p^{m_2}$ and defined as $f([\mathbf{x}_{1,1}]_1, [\mathbf{x}_{1,2}]_2, \dots, [\mathbf{x}_{n,1}]_1, [\mathbf{x}_{n,2}]_2) := [\langle \mathbf{x}, \mathbf{y} \rangle]_T$ where $\mathbf{x} := (\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \dots, \mathbf{x}_{n,1}, \mathbf{x}_{n,2})$ and $\mathbf{y} := (\mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \dots, \mathbf{y}_{n,1}, \mathbf{y}_{n,2})$.

We refer to MIFE for $\mathcal{F}_{m,n,\mathbb{G}}^{\text{MIP}}$ and $\mathcal{F}_{m_1, m_2, n, \mathbb{G}}^{\text{MGIP}}$ as IP-MIFE and mixed-group IP-MIFE, respectively. We require mixed-group IP-MIFE to satisfy the standard function-hiding security in Definition 2.1.

$\text{gSetup}(1^\lambda) \rightarrow \text{gPP}, \text{gMSK}$ $\text{miPP}, \text{miMSK} \leftarrow \text{miSetup}(1^\lambda), (\text{iPP}_1, \text{iMSK}_1), \dots, (\text{iPP}_n, \text{iMSK}_n) \leftarrow \text{iSetup}(1^\lambda)$ $\text{gPP} := (\text{miPP}, \text{iPP}_1, \dots, \text{iPP}_n), \text{gMSK} := (\text{miMSK}, \text{iMSK}_1, \dots, \text{iMSK}_n)$
$\text{gEnc}(\text{MSK}, i, ([\mathbf{x}_{i,1}]_1, [\mathbf{x}_{i,2}]_2)) \rightarrow \text{gCT}_i$ $\mathbf{z} \leftarrow \mathbb{Z}_p^k, \tilde{\mathbf{x}}_{i,1} := (\mathbf{x}_{i,1}, 0^{m_2}, \mathbf{z}, 0) \in \mathbb{Z}_p^{m_1+m_2+k+1}, \tilde{\mathbf{x}}_{i,2} := (\mathbf{x}_{i,2}, -\mathbf{z}, 0) \in \mathbb{Z}_p^{m_2+k+1}$ $\text{miCT}_i \leftarrow \text{miEnc}(\text{miMSK}, i, [\tilde{\mathbf{x}}_{i,1}]_1), \text{iCT}_i \leftarrow \text{iEnc}(\text{iMSK}_i, [\tilde{\mathbf{x}}_{i,2}]_2), \text{gCT}_i := (\text{miCT}_i, \text{iCT}_i)$
$\text{gKeyGen}(\text{MSK}, \{[\mathbf{y}_{i,1}]_2, [\mathbf{y}_{i,2}]_1\}_{i \in [n]}) \rightarrow \text{gSK}$ $\mathbf{a} \leftarrow \mathbb{Z}_p^k, \tilde{\mathbf{y}}_{i,1} := (\mathbf{y}_{i,1}, 0^{m_2}, \mathbf{a}, 0) \in \mathbb{Z}_p^{m_1+m_2+k+1}, \tilde{\mathbf{y}}_{i,2} := (\mathbf{y}_{i,2}, \mathbf{a}, 0) \in \mathbb{Z}_p^{m_2+k+1}, \tilde{\mathbf{y}} := (\tilde{\mathbf{y}}_{1,1}, \dots, \tilde{\mathbf{y}}_{n,1})$ $\text{miSK} \leftarrow \text{miKeyGen}(\text{miMSK}, [\tilde{\mathbf{y}}]_2), \text{iSK}_i \leftarrow \text{iKeyGen}(\text{iMSK}_i, [\tilde{\mathbf{y}}_{i,2}]_1), \text{gSK} := (\text{miSK}, \{\text{iSK}_i\}_{i \in [n]})$
$\text{gDec}(\text{gCT}_1, \dots, \text{gCT}_n, \text{gSK}) \rightarrow z$ $\text{Outputs } \text{miDec}(\text{miCT}_1, \dots, \text{miCT}_n, \text{miSK}) \prod_{i \in [n]} \text{iDec}(\text{iCT}_i, \text{iSK}_i)$

Fig. 2. Our mixed-group IP-MIFE scheme.

4.1 Construction

Let $\mathcal{F}_{m, \mathbb{G}}^{\text{IP}'}$ be a function class defined the same as $\mathcal{F}_{m, \mathbb{G}}^{\text{IP}}$ in Definition 3.1 except that G_1 and G_2 are switched, that is, each $f : G_T^m \rightarrow G_T$ is specified by $[\mathbf{y}]_1$. We construct a function-hiding MIFE scheme for $\mathcal{F}_{m_1, m_2, n, \mathbb{G}}^{\text{MGIP}}$ from a function-hiding MIFE scheme for $\mathcal{F}_{m_1+m_2+k+1, n, \mathbb{G}}^{\text{MIP}}$ and function-hiding FE scheme for $\mathcal{F}_{m_2+k+1, \mathbb{G}}^{\text{IP}'}$ in a generic way. Note that k is a parameter for the MDDH assumption. A function-hiding MIFE scheme for $\mathcal{F}_{m, n, \mathbb{G}}^{\text{MIP}}$ based on MDDH is easily obtained from a function-hiding multi-input IPFE schemes in [4, 17, 30]. This is since these schemes in the literatures work even if input vectors for Enc and KeyGen consist of group elements, and Dec first obtains decryption values on the exponent of a target-group generator and then computes its discrete log.

Let $\text{miFE} = (\text{miSetup}, \text{miEnc}, \text{miKeyGen}, \text{miDec})$ be a function-hiding MIFE scheme for $\mathcal{F}_{m_1+m_2+k+1, n, \mathbb{G}}^{\text{MIP}}$ and $\text{iFE} = (\text{iSetup}, \text{iEnc}, \text{iKeyGen}, \text{iDec})$ be a function-hiding FE scheme for $\mathcal{F}_{m_2+k+1, \mathbb{G}}^{\text{IP}'}$. Then, our function-hiding MIFE scheme $\text{gFE} = (\text{gSetup}, \text{gEnc}, \text{gKeyGen}, \text{gDec})$ for $\mathcal{F}_{m_1, m_2, n, \mathbb{G}}^{\text{MGIP}}$ is constructed as shown in Fig. 2.

Correctness. Due to the correctness of miFE and iFE , gDec outputs

$$\left[\sum_{i \in [n]} (\langle \tilde{\mathbf{x}}_{i,1}, \tilde{\mathbf{y}}_{i,1} \rangle + \langle \tilde{\mathbf{x}}_{i,2}, \tilde{\mathbf{y}}_{i,2} \rangle) \right]_T = \left[\sum_{i \in [n]} (\langle \mathbf{x}_{i,1}, \mathbf{y}_{i,1} \rangle + \langle \mathbf{x}_{i,2}, \mathbf{y}_{i,2} \rangle) \right]_T.$$

For security, we have the following theorem.

Theorem 4.1. *If miFE and iFE are function-hiding, and the bilateral MDDH assumption holds in \mathbb{G} , then gFE is function-hiding. More precisely, for all PPT adversaries \mathcal{A} , there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ such that*

$$\text{Adv}_{\mathcal{A}, \text{fh}}^{\text{gFE}}(\lambda) \leq (4q_{\text{CT}} + 1) \text{Adv}_{\mathcal{B}_1, \text{fh}}^{\text{miFE}}(\lambda) + n(4q_{\text{CT}} + 1) \text{Adv}_{\mathcal{B}_2, \text{fh}}^{\text{iFE}}(\lambda) + 4nq_{\text{CT}} \text{Adv}_{\mathcal{B}_3}^{\text{bi-}\mathcal{D}_k\text{-MDDH}}(\lambda).$$

Due to space constraints, the proof is provided in the full version.

$\begin{aligned} & \text{qSetup}(1^\lambda) \rightarrow \text{qPP}, \text{qMSK} \\ & \mathbb{G} \leftarrow \mathcal{G}_{\text{BG}}(1^\lambda), w_{1,1}, w_{1,2}, w_{2,1}, w_{2,2}, u_1, u_2, v_1, v_2 \leftarrow \mathbb{Z}_p \\ & \text{pPP}, \text{pMSK} \leftarrow \text{pSetup}(1^\lambda), \text{iPP}, \text{iMSK} \leftarrow \text{iSetup}(1^\lambda), \text{gPP}, \text{gMSK} \leftarrow \text{gSetup}(1^\lambda) \\ & \text{qPP} := (\mathbb{G}, \text{pPP}, \text{iPP}, \text{gPP}), \text{qMSK} := (\{w_{i,j}\}_{i,j \in [2]}, \{u_i, v_i\}_{i \in [2]}, \text{pMSK}, \text{iMSK}, \text{gMSK}) \end{aligned}$ $\begin{aligned} & \text{qEnc}(\text{qMSK}, i, x_i) \rightarrow \text{qCT}_i \\ & s, \tilde{s}, r, t, L \leftarrow \mathbb{Z}_p, \mathbf{l} := \mathbf{e}_{i/2} \otimes (1, L) \in \mathbb{Z}_p^4, \tilde{\mathbf{l}} := \mathbf{e}_{i/2} \otimes (L, -1) \in \mathbb{Z}_p^4 \\ & \mathbf{b} := (x_i, 0, sw_{1,i}, sw_{2,i}, u_i, t, 0, 0) \in \mathbb{Z}_p^8, \tilde{\mathbf{b}} := (x_i, 0, \tilde{s}\mathbf{e}_{i/2}, r, v_i, 0, 0) \in \mathbb{Z}_p^8 \\ & \mathbf{d} := (s, 0) \in \mathbb{Z}_p^2, \tilde{\mathbf{d}} := (\tilde{s}, 0) \in \mathbb{Z}_p^2, \mathbf{f} := (r, t, 0, 0) \in \mathbb{Z}_p^4, h := 0 \\ & \text{pCT}_i \leftarrow \text{pEnc}(\text{pMSK}, (1, [\mathbf{b}]_1)), \text{pSK}_i \leftarrow \text{pKeyGen}(\text{pMSK}, (\tilde{\mathbf{l}}, [\tilde{\mathbf{b}}]_2)) \\ & \text{iCT}_i \leftarrow \text{iEnc}(\text{iMSK}, [\mathbf{d}]_1), \text{iSK}_i \leftarrow \text{iKeyGen}(\text{iMSK}, [\tilde{\mathbf{d}}]_2) \\ & \text{gCT}_i \leftarrow \text{gEnc}(\text{gMSK}, i, ([\mathbf{f}]_1, [h]_2)), \text{qCT}_i := (\text{pCT}_i, \text{pSK}_i, \text{iCT}_i, \text{iSK}_i, \text{gCT}_i) \end{aligned}$ $\begin{aligned} & \text{qKeyGen}(\text{qMSK}, \mathbf{c} = \{c_{\mu,\nu}\}_{\mu,\nu \in [2]}) \rightarrow \text{qSK} \\ & \tilde{\mathbf{i}} := \left(\sum_{\mu \in [2]} c_{i,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,i} v_\mu, 0, 0 \right) \in \mathbb{Z}_p^4, \tilde{h}_i := 0, \sigma_{i,\theta} := c_{i,\theta} w_{i,\theta} \\ & \text{gSK} \leftarrow \text{gKeyGen}(\text{gMSK}, \{[\tilde{\mathbf{i}}]_2, [\tilde{h}_i]_1\}_{i \in [2]}), \text{qSK} := (\mathbf{c}, \text{gSK}, \{\sigma_{i,\theta}\}_{i,\theta \in [2]}) \end{aligned}$ $\begin{aligned} & \text{qDec}(\text{qCT}_1, \text{qCT}_2, \text{qSK}) \rightarrow z \\ & [z_1]_T := \prod_{\mu,\nu \in [2]} \text{pDec}(\text{pCT}_\nu, \text{pSK}_\mu)^{c_{\mu,\nu}}, [z_2]_T := \prod_{i,\theta \in [2]} \text{iDec}(\text{iCT}_\theta, \text{iSK}_i)^{\sigma_{i,\theta}} \\ & [z_3]_T := \text{gDec}(\text{gCT}_1, \text{gCT}_2, \text{gSK}), [z]_T := [z_1 - z_2 - z_3]_T \\ & \text{Searches for } z \text{ within the range of } z \leq 4CX^2 \end{aligned}$

Fig. 3. Our two-input quadratic MIFE scheme.

5 Warm-Up: Two Input Quadratic MIFE

Since our general quadratic MIFE scheme (Sect. 6) is quite complex, we first present a simpler scheme as a warm-up. This scheme is a MIFE scheme for $\mathcal{F}_{1,2,X,C}^{\text{MQF}}$ from the SXDH assumption, that is $m = 1, n = 2$. For ease of exposition, we also restrict the number of ciphertext queries to 2 per slot. The SXDH assumption is captured as the \mathcal{D}_k assumption where \mathcal{D}_k consists of all matrices with the form of $(a, 1)^\top \in \mathbb{Z}_p^2$.

Let pFE = (pSetup, pEnc, pKeyGen, pDec) be an FE scheme for $\mathcal{F}_{4,8,\mathbb{G}}^{\text{PIP}}$ (Definition 3.2), iFE = (iSetup, iEnc, iKeyGen, iDec) be an FE scheme for $\mathcal{F}_{2,\mathbb{G}}^{\text{IP}}$ (Definition 3.1), and gFE = (gSetup, gEnc, gKeyGen, gDec) be an FE scheme for $\mathcal{F}_{4,1,2,\mathbb{G}}^{\text{MGIP}}$ (Definition 4.2). The warm-up scheme qFE = (qSetup, qEnc, qKeyGen, qDec) is constructed from pFE, iFE, and gFE as shown in Fig. 3. Since gFE cannot be instantiated from SXDH, the warm-up scheme needs an additional assumption such as XDLIN (bilateral 2-Lin).

Correctness. Let $s_i, \tilde{s}_i, r_i, t_i, \mathbf{l}_i, \tilde{\mathbf{l}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i$ for $i \in [2]$ be random elements used to generate qCT_i . Observe that $\langle \mathbf{l}_i, \tilde{\mathbf{l}}_i \rangle = 0$ for all $i, I \in [2]$, and thus $\text{pDec}(\text{pCT}_i, \text{pSK}_I) = \langle \mathbf{b}_i, \tilde{\mathbf{b}}_I \rangle$. Due to the correctness of pFE, iFE, gFE, we have

$$z_1 = \sum_{\mu, \nu \in [2]} c_{\mu, \nu} (x_\mu x_\nu + s_\nu \tilde{s}_\mu w_{\mu, \nu} + r_\mu u_\nu + t_\nu v_\mu)$$

$$z_2 = \sum_{\mu, \nu \in [2]} c_{\mu, \nu} s_\nu \tilde{s}_\mu w_{\mu, \nu}, \quad z_3 = \sum_{\mu, \nu \in [2]} c_{\mu, \nu} (r_\mu u_\nu + t_\nu v_\mu).$$

Hence, we have $z = \sum_{\mu, \nu \in [2]} c_{\mu, \nu} x_\mu x_\nu$.

5.1 Multi-input IPFE Scheme for Security Analysis

Before going to the security analysis of our quadratic MIFE scheme, we introduce a message-hiding IP-MIFE scheme, i.e. an MIFE scheme for $\mathcal{F}_{m, n, \mathbb{G}}^{\text{MIP}}$, denoted by $\text{miFE} = (\text{miSetup}, \text{miEnc}, \text{miKeyGen}, \text{miDec})$ that we use for the security proof. The scheme is obtained by applying the conversion of single to multi-input IPFE by Abdalla *et al.* [4, Sec. 4.1], to the single-input IPFE scheme by Abdalla *et al.* [3, Sec. 5]. The resulting scheme satisfies the message-hiding security under the DDH assumption. Note that although Abdalla *et al.* considered the conversion in the adaptive setting, it is not hard to see that the conversion works in the selective setting. The original scheme in [3] uses a pairing-free group for the construction, but it is easy to see that their scheme can be similarly built on pairing groups where the SXDH assumption holds. The scheme is described in Fig. 4.

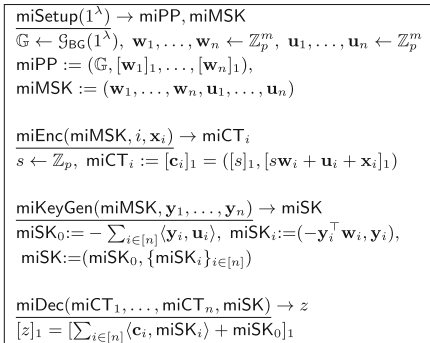


Fig. 4. IP-MIFE scheme by Abdalla *et al.*

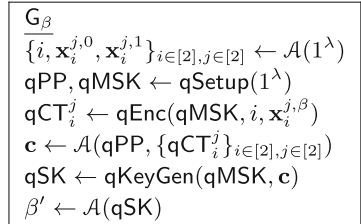


Fig. 5. qFE warmup security game.

5.2 Proof of Security

Theorem 5.1. *If pFE is partially function-hiding, iFE and gFE are function-hiding, and \mathcal{G}_{BG} outputs bilinear groups where the SXDH assumption holds, then qFE is message-hiding as long as $q_{\text{CT}} = 2$ and $q_{\text{SK}} = 1$.*

Proof. For ease of exposition, we prove security in the restricted game where an adversary makes two ciphertext queries per slot and one secret key query. This simplification showcases the basic strategy of the general proof, which is

qCT_1^1 $\mathbf{b}_1^1 := (x_1^{1,\beta}, 0, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}}_1^1 := (x_1^{1,\beta}, 0, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d}_1^1 := (s_1^1, 0), \tilde{\mathbf{d}}_1^1 := (\tilde{s}_1^1, 0)$ $\mathbf{f}_1^1 := (r_1^1, t_1^1, 0, 0), h_1^1 := 0$	qCT_2^1 $\mathbf{b}_2^1 := (x_2^{1,\beta}, 0, s_2^1 w_{1,2}, s_2^1 w_{2,2}, u_2, t_2^1, 0, 0)$ $\tilde{\mathbf{b}}_2^1 := (x_2^{1,\beta}, 0, 0, \tilde{s}_2^1, r_2^1, v_2, 0, 0)$ $\mathbf{d}_2^1 := (s_2^1, 0), \tilde{\mathbf{d}}_2^1 := (\tilde{s}_2^1, 0)$ $\mathbf{f}_2^1 := (r_2^1, t_2^1, 0, 0), h_2^1 := 0$
qCT_1^2 $\mathbf{b}_1^2 := (x_1^{2,\beta}, 0, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, 0, 0)$ $\tilde{\mathbf{b}}_1^2 := (x_1^{2,\beta}, 0, \tilde{s}_1^2, 0, r_1^2, v_1, 0, 0)$ $\mathbf{d}_1^2 := (s_1^2, 0), \tilde{\mathbf{d}}_1^2 := (\tilde{s}_1^2, 0)$ $\mathbf{f}_1^2 := (r_1^2, t_1^2, 0, 0), h_1^2 := 0$	qCT_2^2 $\mathbf{b}_2^2 := (x_2^{2,\beta}, 0, s_2^2 w_{1,2}, s_2^2 w_{2,2}, u_2, t_2^2, 0, 0)$ $\tilde{\mathbf{b}}_2^2 := (x_2^{2,\beta}, 0, 0, \tilde{s}_2^2, r_2^2, v_2, 0, 0)$ $\mathbf{d}_2^2 := (s_2^2, 0), \tilde{\mathbf{d}}_2^2 := (\tilde{s}_2^2, 0)$ $\mathbf{f}_2^2 := (r_2^2, t_2^2, 0, 0), h_2^2 := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, 0, 0)$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, 0, 0)$ $\tilde{h}_2 := 0$

Fig. 6. Vectors in \mathbf{G}_β .

qCT_1^1 $\mathbf{b} := (x_1^{1,0}, \boxed{x_1^{1,1}}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, \boxed{t_1^1 v_1 + x_1^{1,0} x_1^{1,0}})$ $\tilde{\mathbf{b}} := (\boxed{0}, 0, \tilde{s}_1^1, 0, r_1^1, \boxed{0}, 0, \boxed{1})$ $\mathbf{d} := (s_1^1, 0), \tilde{\mathbf{d}} := (s_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, \boxed{t_1^1 v_1}, 0), h = 0$	qCT_2^1 $\mathbf{b} := (x_2^{1,0}, \boxed{x_2^{1,1}}, s_2^1 w_{1,2}, s_2^1 w_{2,2}, u_2, t_2^1, \boxed{t_2^1 v_1}, \boxed{t_2^1 v_1 + x_1^{1,0} x_2^{1,0}})$ $\tilde{\mathbf{b}} := (x_2^{1,0}, 0, 0, \tilde{s}_2^1, r_2^1, v_2, 0, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (s_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, \boxed{t_2^1 v_1}, 0), h = 0$
qCT_1^2 $\mathbf{b} := (x_1^{2,0}, \boxed{x_1^{2,1}}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, \boxed{t_1^2 v_1}, 0)$ $\tilde{\mathbf{b}} := (x_1^{2,0}, 0, \tilde{s}_1^2, 0, r_1^2, \boxed{0}, \boxed{1})$ $\mathbf{d} := (s_1^2, 0), \tilde{\mathbf{d}} := (s_1^2, 0)$ $\mathbf{f} := (r_1^2, t_1^2, \boxed{t_1^2 v_1}, 0), h = 0$	qCT_2^2 $\mathbf{b} := (x_2^{2,0}, \boxed{x_2^{2,1}}, s_2^2 w_{1,2}, s_2^2 w_{2,2}, u_2, t_2^2, \boxed{t_2^2 v_1}, \boxed{t_2^2 v_1 + x_1^{1,0} x_2^{2,0}})$ $\tilde{\mathbf{b}} := (x_2^{2,0}, 0, 0, \tilde{s}_2^2, r_2^2, v_2, 0, 0)$ $\mathbf{d} := (s_2^2, 0), \tilde{\mathbf{d}} := (s_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \boxed{t_2^2 v_1}, 0), h = 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \boxed{c_{2,1} v_2}, \boxed{c_{1,1}}, \boxed{c_{2,1}})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \boxed{c_{2,2} v_2}, \boxed{c_{1,2}}, \boxed{c_{2,2}})$ $\tilde{h}_2 := 0$

Fig. 7. Vectors in \mathbf{H}_1 .

provided in Sect. 6. At a high-level view, our security proof is inspired by that of the IP-MIFE schemes by Abdalla *et al.* [4] in which the first queried ciphertexts of each slot are changed from bit 0 to bit 1 by the information-theoretic property of the one-time pad and the rest of ciphertexts are changed by the security of an IPFE scheme. In our case, the IPFE scheme will instead correspond to the IP-MIFE scheme in Sect. 5.1.

Intuitively, we want to prove $\mathbf{G}_0 \approx_c \mathbf{G}_1$ where \mathbf{G}_β is the message-hiding security game (described in Fig. 5). In \mathbf{G}_β , the vectors in the ciphertexts and the secret key that the adversary obtains are defined as Fig. 6. We introduce a series of hybrid games, $\mathbf{H}_1, \dots, \mathbf{H}_{15}$, and prove $\mathbf{G}_0 \approx_c \mathbf{H}_1 \approx_c \dots \approx_c \mathbf{H}_{15} \approx_c \mathbf{G}_1$. In each hybrid game, the vectors for generating the ciphertexts and the secret keys are changed from \mathbf{G}_0 , which is shown in Fig. 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 and 21. We frame the parts that are changed from the previous game by a box and sometimes denote the parts that are not changed by --- .

$\mathbf{G}_0 \approx_c \mathbf{H}_1$. We can justify this indistinguishability by the (partially) function-hiding property of pFE and gFE. For all $i, j, I, J \in [2]$, we can see that $\langle \mathbf{b}_i^j, \tilde{\mathbf{b}}_I^J \rangle$

qCT_1^1 $\tilde{v}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, \boxed{\tilde{v}_1^1} + x_1^{1,0} x_1^{1,0})$ $\tilde{\mathbf{b}} := (0, 0, \tilde{s}_1^1, 0, r_1^1, 0, 0, 1)$ $\mathbf{d} := (s_1^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, \boxed{\tilde{v}_1^1}, 0), h := 0$	qCT_2^1 $\tilde{v}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (x_2^{1,0}, x_2^{1,1}, s_2^1 w_{1,2}, s_2^1 w_{2,2}, u_2, t_2^1, \boxed{\tilde{v}_2^1}, \boxed{\tilde{v}_2^1} + x_1^{1,0} x_2^{1,0})$ $\tilde{\mathbf{b}} := (x_2^{1,0}, 0, 0, \tilde{s}_2^1, r_2^1, v_2, 0, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, \boxed{\tilde{v}_2^1}, 0), h := 0$
qCT_1^2 $\tilde{v}_1^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (x_1^{2,0}, x_1^{2,1}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, \boxed{\tilde{v}_1^2}, 0)$ $\tilde{\mathbf{b}} := (x_1^{2,0}, 0, \tilde{s}_1^2, 0, r_1^2, 0, 1, 0)$ $\mathbf{d} := (s_1^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^2, 0)$ $\mathbf{f} := (r_1^2, t_1^2, \boxed{\tilde{v}_1^2}, 0), h := 0$	qCT_2^2 $\tilde{v}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (x_2^{2,0}, x_2^{2,1}, s_2^2 w_{1,2}, s_2^2 w_{2,2}, u_2, t_2^2, \boxed{\tilde{v}_2^2}, \boxed{\tilde{v}_2^2} + x_1^{1,0} x_2^{2,0})$ $\tilde{\mathbf{b}} := (x_2^{2,0}, 0, 0, \tilde{s}_2^2, r_2^2, v_2, 0, 0)$ $\mathbf{d} := (s_2^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \boxed{\tilde{v}_2^2}, 0), h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, c_{2,1} v_2, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, c_{2,2} v_2, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

 Fig. 8. Vectors in \mathbf{H}_2 .

qCT_1^1 $\tilde{v}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, 0, \tilde{v}_1^1 + \boxed{x_1^{1,1} x_1^{1,1}})$ $\tilde{\mathbf{b}} := (-, 0, 1)$ $\mathbf{d} := (s_1^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, \tilde{v}_1^1 + \boxed{x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}}, 0), h := 0$	qCT_2^1 $\tilde{v}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{v}_2^1 + \boxed{x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}}, \tilde{v}_2^1 + \boxed{x_1^{1,1} x_2^{1,1}})$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, \tilde{v}_2^1 + \boxed{x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}}, 0), h := 0$
qCT_1^2 $\tilde{v}_1^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{v}_1^2 + \boxed{x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}}, 0)$ $\tilde{\mathbf{b}} := (-, 1, 0)$ $\mathbf{d} := (s_1^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \tilde{v}_1^2 + \boxed{x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}}, 0), h := 0$	qCT_2^2 $\tilde{v}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{v}_2^2 + \boxed{x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}}, \tilde{v}_2^2 + \boxed{x_1^{1,1} x_2^{2,1}})$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \tilde{v}_2^2 + \boxed{x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}}, 0), h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, c_{2,1} v_2, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, c_{2,2} v_2, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

 Fig. 9. Vectors in \mathbf{H}_3 .

in \mathbf{G}_0 and that in \mathbf{H}_1 are equal unless $i = I$ and $j \neq J$. Recall that $\langle \mathbf{l}_i^j, \tilde{\mathbf{l}}_I^j \rangle \neq 0$ with overwhelming probability if $i = I$ and $j \neq J$, since L is chosen from the exponentially large space, \mathbb{Z}_p . Hence, the indistinguishability of $\{\mathbf{b}, \tilde{\mathbf{b}}\}$ between \mathbf{G}_0 and \mathbf{H}_1 is implied by the partially function-hiding property of pFE.

Similarly, for all $i, j \in [2]$, $\langle \mathbf{f}_i^j, \tilde{\mathbf{f}}_i \rangle$ in \mathbf{G}_0 and that in \mathbf{H}_1 are equal, which implies, for all $j_1, j_2 \in [2]$, $\sum_{i \in [2]} (\langle \mathbf{f}_i^{j_1}, \tilde{\mathbf{f}}_i \rangle + h_i^{j_1} \tilde{h}_i)$ in \mathbf{G}_0 and that in \mathbf{H}_1 are equal. Thus, the indistinguishability of $\{\mathbf{f}, \tilde{\mathbf{f}}\}$ between \mathbf{G}_0 and \mathbf{H}_1 is implied by the function-hiding property of gFE.

$\mathbf{H}_1 \approx_c \mathbf{H}_2$. We can justify this indistinguishability by the SXDH assumption, which implies $(\mathbb{G}, [\mathbf{t}]_1, [v_1 \mathbf{t}]_1) \approx_c (\mathbb{G}, [\mathbf{t}]_1, [\tilde{\mathbf{v}}]_1)$ where $\mathbb{G} \leftarrow \mathcal{G}_{\text{BG}}(1^\lambda)$, $\mathbf{t} = \{t_i^j\}_{i,j \in [2]}$, $\tilde{\mathbf{v}} = \{\tilde{v}_i^j\}_{i,j \in [2]} \leftarrow \mathbb{Z}_p^4$, $v_1 \leftarrow \mathbb{Z}_p$.

$\underline{\text{qCT}}_1^1$ $\mathbf{b} := (-, 0, \boxed{t_1^1 v_1} + x_1^{1,1} x_1^{1,1})$ $\tilde{\mathbf{b}} := (-, 0, 1)$ $\mathbf{d} := (s_1^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, \boxed{t_1^1 v_1} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	$\underline{\text{qCT}}_1^2$ $\mathbf{b} := (-, \boxed{t_2^1 v_1} + x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, \boxed{t_2^1 v_1} + x_1^{1,1} x_2^{1,1})$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, \boxed{t_2^1 v_1} + x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0), h := 0$
$\underline{\text{qCT}}_2^1$ $\mathbf{b} := (-, \boxed{t_1^2 v_1} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$ $\tilde{\mathbf{b}} := (-, 1, 0)$ $\mathbf{d} := (s_1^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^2, 0)$ $\mathbf{f} := (r_1^2, t_1^2, \boxed{t_1^2 v_1} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	$\underline{\text{qCT}}_2^2$ $\mathbf{b} := (-, \boxed{t_2^2 v_1} + x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, \boxed{t_2^2 v_1} + x_1^{1,1} x_2^{2,1})$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \boxed{t_2^2 v_1} + x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0), h := 0$
$\underline{\text{qSK}}$ $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, c_{2,1} v_2, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, c_{2,2} v_2, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 10. Vectors in H_4 .

$\underline{\text{qCT}}_1^1$ $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, \boxed{0})$ $\tilde{\mathbf{b}} := (0, \boxed{x_1^{1,1}}, \tilde{s}_1^1, 0, r_1^1, \boxed{v_1}, \boxed{0}, \boxed{0})$ $\mathbf{d} := (s_1^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, \boxed{t_1^1 v_1} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	$\underline{\text{qCT}}_1^2$ $\mathbf{b} := (-, \boxed{t_2^1 v_1} + x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, \boxed{0})$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, \boxed{t_2^1 v_1} + x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0), h := 0$
$\underline{\text{qCT}}_2^1$ $\mathbf{b} := (x_1^{2,0}, x_1^{2,1}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, \boxed{t_1^2 v_1} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$ $\tilde{\mathbf{b}} := (x_1^{2,0}, 0, \tilde{s}_1^2, 0, r_1^2, \boxed{v_1}, 1, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_1^2, t_1^2, \boxed{t_1^2 v_1} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	$\underline{\text{qCT}}_2^2$ $\mathbf{b} := (-, \boxed{t_2^2 v_1} + x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, \boxed{0})$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \boxed{t_2^2 v_1} + x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0), h := 0$
$\underline{\text{qSK}}$ $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 11. Vectors in H_5 .

$H_2 \approx H_3$. These hybrid games are information-theoretically equivalent. This can be confirmed by setting $\tilde{v}_i^j := \begin{cases} v_i^{t,j} + x_1^{1,1} x_i^{1,1} - x_1^{1,0} x_i^{1,0} & (i = 1) \\ v_i^{t,j} + x_1^{1,1} x_i^{j,1} - x_1^{1,0} x_i^{j,0} & (i = 2) \end{cases}$ where $v_i^{t,j} \leftarrow \mathbb{Z}_p$.

$H_3 \approx_c H_4$. We can justify this indistinguishability by the SXDH assumption, and the indistinguishability can be shown similarly to that between H_1 and H_2 .

$H_4 \approx_c H_5$. We can justify this indistinguishability by the (partially) function-hiding property of pFE and gFE, similarly to the case of $G_0 \approx_c H_1$.

$H_5 \approx_c H_6$. We can justify this indistinguishability by the (partially) function-hiding property of pFE, iFE, and gFE, similarly to the case of $G_0 \approx_c H_1$. Note that here we also need to consider iFE since $\{\mathbf{d}, \tilde{\mathbf{d}}\}$ is also changed, but it is easy to see that, for all $i, j, I, J \in [2]$, $\langle \mathbf{d}_i^j, \tilde{\mathbf{d}}_I^J \rangle$ in H_5 and that in H_6 are equal.

$H_6 \approx_c H_7$. We can justify this indistinguishability by the SXDH assumption, which implies $(\mathbb{G}, [\mathbf{s}]_1, [\tilde{s}_1^2 \mathbf{s}]_1) \approx_c (\mathbb{G}, [\mathbf{s}]_1, [\tilde{\mathbf{s}}]_1)$ and $(\mathbb{G}, [\mathbf{u}]_1, [r_1^2 \mathbf{u}]_1) \approx_c (\mathbb{G}, [\mathbf{u}]_1, [\tilde{\mathbf{u}}]_1)$ where $\mathbb{G} \leftarrow \mathcal{G}_{\text{BG}}(1^\lambda), \mathbf{s} = \{s_i^j\}_{i,j \in [2]}, \tilde{\mathbf{s}} = \{\tilde{s}_i^j\}_{i,j \in [2]} \leftarrow \mathbb{Z}_p^4, \tilde{s}_1^2 \leftarrow \mathbb{Z}_p, \mathbf{u} = \{u_i\}_{i \in [2]}, \tilde{\mathbf{u}} = \{\tilde{u}_i\}_{i \in [2]} \leftarrow \mathbb{Z}_p^2, r_1^2 \leftarrow \mathbb{Z}_p$.

qCT_1^1 $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1, \tilde{s}_1^1)$, $\tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$, $h := 0$	qCT_2^1 $\mathbf{b} := (-, \sqrt{s_2^2 w_{1,2} + r_2^1 u_2 + x_2^{2,0} x_2^{1,0}} + x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1, \tilde{s}_2^1)$, $\tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_2^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0)$, $h := 0$
qCT_1^2 $\mathbf{b} := (-, \sqrt{s_1^2 w_{1,1} + r_1^2 u_1 + x_1^{2,0} x_1^{2,0}} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$ $\tilde{\mathbf{b}} := (0, 0, 0, 0, 0, v_1, 1, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1, \tilde{s}_1^1)$, $\tilde{\mathbf{d}} := (0, 1)$ $\mathbf{f} := (0, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$, $h := 1$	qCT_2^2 $\mathbf{b} := (-, \sqrt{s_2^2 w_{1,2} + r_2^1 u_2 + x_2^{2,0} x_2^{2,0}} + x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1, \tilde{s}_2^1)$, $\tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_2^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0)$, $h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := r_1^1 \sum_{\mu \in [2]} c_{1,\mu} u_\mu$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 12. Vectors in H_6 .

Additional sampling for qMSK	
$\tilde{u}_1, \tilde{u}_2 \leftarrow \mathbb{Z}_p$	
qCT_1^1 $\tilde{s}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1)$, $\tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$, $h := 0$	qCT_2^1 $\tilde{s}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \sqrt{s_2^1 w_{1,2} + \tilde{u}_2} + x_1^{2,0} x_2^{1,0} + x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1)$, $\tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_2^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0)$, $h := 0$
qCT_1^2 $\tilde{s}_1^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \sqrt{s_1^2 w_{1,1} + \tilde{u}_1} + x_2^{2,0} x_1^{2,0} + x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$ $\tilde{\mathbf{b}} := (-, 1, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1)$, $\tilde{\mathbf{d}} := (0, 1)$ $\mathbf{f} := (0, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0)$, $h := 1$	qCT_2^2 $\tilde{s}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \sqrt{s_2^2 w_{1,2} + \tilde{u}_2} + x_2^{2,0} x_2^{2,0} + x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1)$, $\tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_2^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0)$, $h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := \sum_{\mu \in [2]} c_{1,\mu} \tilde{u}_\mu$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 13. Vectors in H_7 .

$H_7 \approx_c H_8$. We can justify this indistinguishability by the message-hiding property of miFE. First, we prove that, for all $j \in [2]$, we have

$$\begin{aligned} & c_{1,1}(x_1^{2,0} x_1^{2,0} - x_1^{1,0} x_1^{1,0}) + c_{1,2}(x_1^{2,0} x_2^{j,0} - x_1^{1,0} x_2^{j,0}) \\ &= c_{1,1}(x_1^{2,1} x_1^{2,1} - x_1^{1,1} x_1^{1,1}) + c_{1,2}(x_1^{2,1} x_2^{j,1} - x_1^{1,1} x_2^{j,1}). \end{aligned} \quad (5.1)$$

Due to the game condition defined in Definition 2.1, the queries by the adversary satisfy

$$\sum_{i, \theta \in [2]} c_{i, \theta} x_i^{f(i), 0} x_\theta^{f(\theta), 0} = \sum_{i, \theta \in [2]} c_{i, \theta} x_i^{f(i), 1} x_\theta^{f(\theta), 1} \quad (5.2)$$

$$\sum_{i, \theta \in [2]} c_{i, \theta} x_i^{g(i), 0} x_\theta^{g(\theta), 0} = \sum_{i, \theta \in [2]} c_{i, \theta} x_i^{g(i), 1} x_\theta^{g(\theta), 1} \quad (5.3)$$

where $f(i) = \begin{cases} 2 & (i = 1) \\ j & (i = 2) \end{cases}$, $g(i) = \begin{cases} 1 & (i = 1) \\ j & (i = 2) \end{cases}$. Note that Eq. (5.2) represents the restriction $f(x_1^{2,0}, x_2^{j,0}) = f(x_1^{2,1}, x_2^{j,1})$, and Eq. (5.3) represents the

Additional sampling for qMSK	
$\ddot{u}_1, \ddot{u}_2 \leftarrow \mathbb{Z}_p$	
qCT_1^1 $\ddot{s}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, \ddot{s}_1^1), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	qCT_2^1 $\ddot{s}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \ddot{s}_2^1 w_{1,2} + \ddot{u}_2 + \boxed{x_1^{2,1} x_2^{1,1}}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \ddot{s}_2^1), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0), h := 0$
qCT_1^2 $\ddot{s}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \ddot{s}_2^1 w_{1,1} + \ddot{u}_1 + \boxed{x_1^{2,1} x_2^{1,1}}, 0)$ $\tilde{\mathbf{b}} := (-, 1, 0)$ $\mathbf{d} := (s_2^1, \ddot{s}_2^1), \tilde{\mathbf{d}} := (0, 1)$ $\mathbf{f} := (0, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0), h := 1$	qCT_2^2 $\ddot{s}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \ddot{s}_2^2 w_{1,2} + \ddot{u}_2 + \boxed{x_1^{2,1} x_2^{2,1}}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^2, \ddot{s}_2^2), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0), h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := \sum_{\mu \in [2]} c_{1,\mu} \ddot{u}_\mu$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 14. Vectors in \mathbf{H}_8 .

restriction $f(x_1^{1,0}, x_2^{j,0}) = f(x_1^{1,1}, x_2^{j,1})$. Equation (5.2)–Eq. (5.3) implies Eq. (5.1) by reflecting the fact that $c_{2,1} = 0$, which is defined in Definition 2.2.

Thanks to the message-hiding property of 2-slot miFE and Eq. (5.1), we have

$$\{\text{miPP}, \text{miCT}_1^{1,0}, \text{miCT}_2^{1,0}, \text{miCT}_2^{2,0}, \text{miSK}\} \approx_c \{\text{miPP}, \text{miCT}_1^{1,1}, \text{miCT}_2^{1,1}, \text{miCT}_2^{2,1}, \text{miSK}\}$$

where

$$\begin{aligned} \text{miPP} &= (\mathbb{G}, [w_{1,1}]_1, [w_{1,2}]_1) \\ \text{miCT}_1^{1,\beta} &= ([\ddot{s}_1^1]_1, [\ddot{s}_1^2 w_{1,1} + \ddot{u}_1 + x_1^{2,\beta} x_2^{1,\beta} - x_1^{1,\beta} x_2^{1,\beta}]_1) \\ \text{miCT}_2^{j,\beta} &= ([\ddot{s}_2^j]_1, [\ddot{s}_2^j w_{1,2} + \ddot{u}_2 + \underbrace{x_1^{2,\beta} x_2^{j,\beta} - x_1^{1,\beta} x_2^{j,\beta}}_{\text{message vectors}}]_1) \\ \text{miSK} &= \left(\sum_{\mu \in [2]} c_{1,\mu} \ddot{u}_\mu, -c_{1,1} w_{1,1}, -c_{1,2} w_{1,2}, \underbrace{c_{1,1}, c_{1,2}}_{\text{key vector}} \right). \end{aligned}$$

Roughly speaking, $[\mathbf{b}]_1$ in $\text{qCT}_1^2, \text{qCT}_2^1, \text{qCT}_2^2$ is simulatable from $\text{miCT}_1^{1,\beta}, \text{miCT}_2^{1,\beta}, \text{miCT}_2^{2,\beta}$, respectively, and $[\tilde{h}_1]_1$ in qSK is simulatable from miSK , and the case of $\beta = 0$ corresponds to \mathbf{H}_7 and $\beta = 1$ corresponds to \mathbf{H}_8 .

$\mathbf{H}_8 \approx_c \mathbf{H}_9$. We can justify this indistinguishability by the SXDH assumption similarly to the case of $\mathbf{H}_6 \approx_c \mathbf{H}_7$.

$\mathbf{H}_9 \approx_c \mathbf{H}_{10}$. We can justify this indistinguishability by the (partially) function-hiding property of pFE, iFE, and gFE, similarly to the case of $\mathbf{G}_5 \approx_c \mathbf{H}_6$. At this point, all ciphertexts for slot 1 are changed from encryption of 0-side to that of 1-side.

qCT_1^1 $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, \boxed{s_1^2 s_1^1}), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	qCT_2^1 $\mathbf{b} := (-\boxed{s_2^2 s_1^1 w_{1,2}} + \boxed{r_1^2 u_2} + x_1^{2,1} x_2^{1,1}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \boxed{s_2^2 s_1^1}), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0), h := 0$
qCT_1^2 $\mathbf{b} := (x_1^{2,0}, x_1^{2,1}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, \boxed{s_1^2 s_1^1 w_{1,1}} + \boxed{r_1^2 u_1} + x_1^{2,1} x_1^{2,1}, 0)$ $\tilde{\mathbf{b}} := (0, 0, \tilde{s}_1^2, 0, 0, v_1, 1, 0)$ $\mathbf{d} := (s_1^2, \boxed{s_1^2 s_1^1}), \tilde{\mathbf{d}} := (0, 1)$ $\mathbf{f} := (0, t_1^2, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 1$	qCT_2^2 $\mathbf{b} := (-\boxed{s_2^2 s_1^1 w_{1,2}} + \boxed{r_1^2 u_2} + x_1^{2,1} x_2^{2,1}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^2, \boxed{s_2^2 s_1^1}), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0), h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := r_1^2 \sum_{\mu \in [2]} c_{1,\mu} u_\mu$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 15. Vectors in H_9 .

qCT_1^1 $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := 0$	qCT_2^1 $\mathbf{b} := (x_2^{1,0}, x_2^{1,1}, s_2^1 w_{1,2}, s_2^1 w_{2,2}, u_2, t_2^1, \boxed{0}, 0)$ $\tilde{\mathbf{b}} := (x_2^{1,0}, 0, 0, \tilde{s}_2^1, r_2^1, v_2, 0, 0)$ $\mathbf{d} := (s_2^1, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, 0), h := 0$
qCT_1^2 $\mathbf{b} := (x_1^{2,0}, x_1^{2,1}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, \boxed{0}, 0)$ $\tilde{\mathbf{b}} := (0, \boxed{x_1^{2,1}}, \boxed{\tilde{s}_1^2}, 0, \boxed{r_1^2}, v_1, \boxed{0}, 0)$ $\mathbf{d} := (s_1^2, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_1^2, \boxed{0})$ $\mathbf{f} := (r_2^2, t_2^2, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, 0), h := \boxed{0}$	qCT_2^2 $\mathbf{b} := (x_2^{2,0}, x_2^{2,1}, s_2^2 w_{1,2}, s_2^2 w_{2,2}, u_2, t_2^2, \boxed{0}, 0)$ $\tilde{\mathbf{b}} := (x_2^{2,0}, 0, 0, \tilde{s}_2^2, r_2^2, v_2, 0, 0)$ $\mathbf{d} := (s_2^2, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, 0), h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := \boxed{0}$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := 0$

Fig. 16. Vectors in H_{10} .

$H_{10} \approx_c H_{11}$. As stated above, G_0 to H_{10} are hybrid games for processing the ciphertexts for slot 1. Next, we apply a similar procedure to slot 2. H_{11} in the process for slot 2 corresponds to H_7 in the process for slot 1. That is, $H_{10} \approx_c H_{11}$ can be proven similarly to $G_0 \approx_c H_7$.

$H_{11} \approx_c H_{12}$. This indistinguishability can be prove similarly to the case of $H_7 \approx_c H_8$, but we need an additional tweak in this case. First, we prove that, for all $j \in [2]$, we have

$$\begin{aligned}
& c_{2,1}(x_2^{2,0} x_1^{j,0} - x_2^{1,0} x_1^{j,0}) + c_{2,2}(x_2^{2,0} x_2^{j,0} - x_2^{1,0} x_2^{j,0}) + c_{1,2}(x_1^{1,0} x_2^{2,0} - x_1^{1,0} x_2^{1,0}) \\
& = c_{2,1}(x_2^{2,1} x_1^{j,1} - x_2^{1,1} x_1^{j,1}) + c_{2,2}(x_2^{2,1} x_2^{j,1} - x_2^{1,1} x_2^{j,1}) + c_{1,2}(x_1^{1,1} x_2^{2,1} - x_1^{1,1} x_2^{1,1}).
\end{aligned} \tag{5.4}$$

Due to the game condition defined in Definition 2.1, the queries by the adversary satisfy

$$\sum_{i, \theta \in [2]} c_{i, \theta} x_i^{f(i), 0} x_\theta^{f(\theta), 0} = \sum_{i, \theta \in [2]} c_{i, \theta} x_i^{f(i), 1} x_\theta^{f(\theta), 1} \tag{5.5}$$

Additional sampling for qMSK	
$\tilde{u}_1, \tilde{u}_2 \leftarrow \mathbb{Z}_p$	
qCT_1^1 $\tilde{s}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_1^1 w_{2,1} + \tilde{u}_1 + x_2^{2,0} x_1^{1,0} + x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0}), h := 0$	qCT_2^1 $\tilde{s}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, -, 0, 0)$ $\tilde{\mathbf{b}} := (\boxed{0}, \boxed{x_2^{1,1}}, 0, \tilde{s}_2^1, r_1^1, v_2, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := 0$
qCT_1^2 $\tilde{s}_1^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_1^2 w_{2,1} + \tilde{u}_1 + x_2^{2,0} x_1^{2,0} + x_2^{1,1} x_1^{2,1} - x_2^{1,0} x_1^{2,0}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_1^2, \tilde{s}_1^2), \tilde{\mathbf{d}} := (\tilde{s}_1^2, 0)$ $\mathbf{f} := (r_1^2, t_1^2, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{2,1} - x_2^{1,0} x_1^{2,0}), h := 0$	qCT_2^2 $\tilde{s}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, -, \tilde{s}_2^2 w_{2,2} + \tilde{u}_2 + x_2^{2,0} x_2^{2,0} + x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}, 0)$ $\tilde{\mathbf{b}} := (\boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, v_2, \boxed{1}, \boxed{1})$ $\mathbf{d} := (s_2^2, \tilde{s}_2^2), \tilde{\mathbf{d}} := (\boxed{0}, \boxed{1})$ $\mathbf{f} := (\boxed{0}, t_2^2, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := \boxed{1}$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := \sum_{\mu \in [2]} c_{1,\mu} \tilde{u}_\mu$

Fig. 17. Vectors in H_{11} .

Additional sampling for qMSK	
$\tilde{u}_1, \tilde{u}_2 \leftarrow \mathbb{Z}_p$	
qCT_1^1 $\tilde{s}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_1^1 w_{2,1} + \tilde{u}_1 + \boxed{x_2^{2,1} x_1^{1,1}}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0})$ $h := 0$	qCT_2^1 $\tilde{s}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, 0, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := 0$
qCT_1^2 $\tilde{s}_1^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_1^2 w_{2,1} + \tilde{u}_1 + \boxed{x_2^{2,1} x_1^{2,1}}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_1^2, \tilde{s}_1^2), \tilde{\mathbf{d}} := (\tilde{s}_1^2, 0)$ $\mathbf{f} := (r_1^2, t_1^2, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{2,1} - x_2^{1,0} x_1^{2,0})$ $h := 0$	qCT_2^2 $\tilde{s}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_2^2 w_{2,2} + \tilde{u}_2 + \boxed{x_2^{2,1} x_2^{2,1}}, 0)$ $\tilde{\mathbf{b}} := (-, 1, 0)$ $\mathbf{d} := (s_2^2, \tilde{s}_2^2), \tilde{\mathbf{d}} := (0, 1)$ $\mathbf{f} := (0, t_2^2, x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := 1$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := \sum_{\mu \in [2]} c_{1,\mu} \tilde{u}_\mu + c_{1,2} (x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0} - (x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0}))$

Fig. 18. Vectors in H_{12} .

$$\sum_{i, \theta \in [2]} c_{i, \theta} x_i^{g(i), 0} x_\theta^{g(\theta), 0} = \sum_{i, \theta \in [2]} c_{i, \theta} x_i^{g(i), 1} x_\theta^{g(\theta), 1} \quad (5.6)$$

where $f(i) = \begin{cases} 1 & (i = 1) \\ 2 & (i = 2) \end{cases}$, $g(i) = \begin{cases} 1 & (i = 1) \\ 1 & (i = 2) \end{cases}$. Note that Eq. (5.5) represents

the restriction $f(x_1^{1,0}, x_2^{2,0}) = f(x_1^{1,1}, x_2^{2,1})$, and Eq. (5.6) represents the restriction $f(x_1^{1,0}, x_2^{1,0}) = f(x_1^{1,1}, x_2^{1,1})$. Equation (5.5)–Eq. (5.6) implies Eq. (5.4) by reflecting the fact that $c_{2,1} = 0$, which is defined in Definition 2.2.

Thanks to the message-hiding property of 3-slot miFE and Eq. (5.4), we have

$$\begin{aligned} & \{\text{miPP}, \text{miCT}_1^{1,0}, \text{miCT}_1^{2,0}, \text{miCT}_2^{1,0}, \text{miCT}_3^{1,0}, \text{miSK}\} \\ & \approx_c \{\text{miPP}, \text{miCT}_1^{1,1}, \text{miCT}_1^{2,1}, \text{miCT}_2^{1,1}, \text{miCT}_3^{1,1}, \text{miSK}\} \end{aligned}$$

Additional sampling for qMSK	
$\tilde{u}_1, \tilde{u}_2 \leftarrow \mathbb{Z}_p$	
qCT_1^1 $\tilde{s}_1^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_1^1 w_{2,1} + \tilde{u}_1 + x_2^{2,1} x_1^{1,1}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_1^1, \tilde{s}_1^1), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0})$ $h := 0$	qCT_2^1 $\tilde{s}_2^1 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, 0, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_2^1, \tilde{s}_2^1), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, x_2^{2,1} x_2^{1,1} - x_2^{2,0} x_2^{1,0}), h := 0$
qCT_1^2 $\tilde{s}_1^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_1^2 w_{2,1} + \tilde{u}_1 + x_2^{2,1} x_1^{2,1}, 0)$ $\tilde{\mathbf{b}} := (-, 0, 0)$ $\mathbf{d} := (s_1^2, \tilde{s}_1^2), \tilde{\mathbf{d}} := (\tilde{s}_1^2, 0)$ $\mathbf{f} := (r_1^2, t_1^2, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, \boxed{x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0}})$ $h := 0$	qCT_2^2 $\tilde{s}_2^2 \leftarrow \mathbb{Z}_p$ $\mathbf{b} := (-, \tilde{s}_2^2 w_{2,2} + \tilde{u}_2 + x_2^{2,1} x_2^{2,1}, 0)$ $\tilde{\mathbf{b}} := (-, 1, 0)$ $\mathbf{d} := (s_2^2, \tilde{s}_2^2), \tilde{\mathbf{d}} := (0, 1)$ $\mathbf{f} := (0, t_2^2, \boxed{x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := 1$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := \sum_{\mu \in [2]} c_{1,\mu} \tilde{u}_\mu + c_{1,2} \cancel{x_1^{1,1} x_2^{1,1}} - \cancel{x_1^{1,0} x_2^{1,0}} - \cancel{(x_1^{1,1} x_2^{2,1} - x_1^{1,0} x_2^{2,0})}$

Fig. 19. Vectors in \mathbf{H}_{13} .

qCT_1^1 $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, \boxed{0}, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0}), h := 0$	qCT_2^1 $\mathbf{b} := (x_2^{1,0}, x_2^{1,1}, s_2^1 w_{1,2}, s_2^1 w_{2,2}, u_2, t_2^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_2^{1,1}, 0, \tilde{s}_2^1, r_2^1, v_2, 0, 0)$ $\mathbf{d} := (s_2^2, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := 0$
qCT_1^2 $\mathbf{b} := (x_1^{2,0}, x_1^{2,1}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, \boxed{0}, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{2,1}, 0, \tilde{s}_1^2, r_1^2, v_1, 0, 0)$ $\mathbf{d} := (s_1^2, \boxed{0}), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^2, t_1^2, x_1^{1,1} x_1^{1,1} - x_1^{1,0} x_1^{1,0}, x_2^{1,1} x_1^{1,1} - x_2^{1,0} x_1^{1,0}), h := 0$	qCT_2^2 $\mathbf{b} := (x_2^{2,0}, x_2^{2,1}, s_2^2 w_{1,2}, s_2^2 w_{2,2}, u_2, t_2^2, \boxed{0}, 0)$ $\tilde{\mathbf{b}} := (0, \boxed{x_2^{2,1}}, \boxed{\tilde{s}_2^2}, 0, \boxed{r_2^2}, v_2, \boxed{0}, 0)$ $\mathbf{d} := (s_2^2, \boxed{0}), \tilde{\mathbf{d}} := (\boxed{\tilde{s}_2^2}, \boxed{0})$ $\mathbf{f} := (\boxed{r_2^2}, t_2^2, x_1^{1,1} x_2^{1,1} - x_1^{1,0} x_2^{1,0}, x_2^{1,1} x_2^{1,1} - x_2^{1,0} x_2^{1,0}), h := \boxed{0}$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, c_{1,1}, c_{2,1})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, c_{1,2}, c_{2,2})$ $\tilde{h}_2 := \boxed{0}$

Fig. 20. Vectors in \mathbf{H}_{14} .

qCT_1^1 $\mathbf{b} := (x_1^{1,0}, x_1^{1,1}, s_1^1 w_{1,1}, s_1^1 w_{2,1}, u_1, t_1^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{1,1}, \tilde{s}_1^1, 0, r_1^1, v_1, 0, 0)$ $\mathbf{d} := (s_1^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^1, t_1^1, \boxed{0}, \boxed{0}), h := 0$	qCT_2^1 $\mathbf{b} := (x_2^{1,0}, x_2^{1,1}, s_2^1 w_{1,2}, s_2^1 w_{2,2}, u_2, t_2^1, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_2^{1,1}, 0, \tilde{s}_2^1, r_2^1, v_2, 0, 0)$ $\mathbf{d} := (s_2^1, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^1, 0)$ $\mathbf{f} := (r_2^1, t_2^1, \boxed{0}, \boxed{0}), h := 0$
qCT_1^2 $\mathbf{b} := (x_1^{2,0}, x_1^{2,1}, s_1^2 w_{1,1}, s_1^2 w_{2,1}, u_1, t_1^2, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_1^{2,1}, 0, \tilde{s}_1^2, r_1^2, v_1, 0, 0)$ $\mathbf{d} := (s_1^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_1^1, 0)$ $\mathbf{f} := (r_1^2, t_1^2, \boxed{0}, \boxed{0}), h := 0$	qCT_2^2 $\mathbf{b} := (x_2^{2,0}, x_2^{2,1}, s_2^2 w_{1,2}, s_2^2 w_{2,2}, u_2, t_2^2, 0, 0)$ $\tilde{\mathbf{b}} := (0, x_2^{2,1}, \tilde{s}_2^2, 0, r_2^2, v_2, 0, 0)$ $\mathbf{d} := (s_2^2, 0), \tilde{\mathbf{d}} := (\tilde{s}_2^2, 0)$ $\mathbf{f} := (r_2^2, t_2^2, \boxed{0}, \boxed{0}), h := 0$
qSK $\tilde{\mathbf{f}}_1 := (\sum_{\mu \in [2]} c_{1,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,1} v_\mu, \boxed{0}, \boxed{0})$ $\tilde{h}_1 := 0$	$\tilde{\mathbf{f}}_2 := (\sum_{\mu \in [2]} c_{2,\mu} u_\mu, \sum_{\mu \in [2]} c_{\mu,2} v_\mu, \boxed{0}, \boxed{0})$ $\tilde{h}_2 := 0$

Fig. 21. Vectors in \mathbf{H}_{15} .

where

$$\begin{aligned}
 \text{miPP} &= (\mathbb{G}, [w_{2,1}]_1, [w_{2,2}]_1, [w_{2,3}]_1) \\
 \text{miCT}_1^{j,\beta} &= ([s_1^j]_1, [s_1^j w_{2,1} + \ddot{u}_1 + x_2^{2,\beta} x_1^{j,\beta} - x_2^{1,\beta} x_1^{j,\beta}]_1) \\
 \text{miCT}_2^{1,\beta} &= ([s_2^1]_1, [s_2^1 w_{2,2} + \ddot{u}_2 + x_2^{2,\beta} x_2^{2,\beta} - x_2^{1,\beta} x_2^{1,\beta}]_1) \\
 \text{miCT}_3^{1,\beta} &= ([s_3^1]_1, [s_3^1 w_{2,3} + \ddot{u}_3 + \underbrace{x_1^{1,\beta} x_2^{2,\beta} - x_1^{1,\beta} x_2^{1,\beta}}_{\text{message vectors}}]_1) \\
 \text{miSK} &= \left(\sum_{\mu \in [2]} c_{2,\mu} \ddot{u}_\mu + c_{1,2} \ddot{u}_3, -c_{2,1} w_{2,1}, -c_{2,2} w_{2,2}, -c_{1,2} w_{2,3}, \underbrace{c_{2,1}, c_{2,2}, c_{1,2}}_{\text{key vector}} \right).
 \end{aligned}$$

Roughly speaking, $[b]_1$ in $\text{qCT}_1^1, \text{qCT}_2^2$ is simulatable from $\text{miCT}_1^{1,\beta}$, $\text{miCT}_1^{2,\beta}, \text{miCT}_2^{1,\beta}$, respectively, and $[\tilde{h}_2]_1$ in qSK is simulatable from miSK and $\text{miCT}_3^{1,\beta}$. More precisely,

$$\tilde{h}_2 = K_1 - C_1 K_4 - c_{1,2} (C_2 + x_1^{1,0} x_2^{2,0} - x_1^{1,0} x_2^{1,0})$$

where $\text{miCT}_3^{1,\beta} = ([C_1]_1, [C_2]_1)$ and $\text{miSK} = (K_1, \dots, K_7)$. The case of $\beta = 0$ corresponds to H_{11} and $\beta = 1$ corresponds to H_{12} .

$H_{12} \approx_c H_{13}$. We can justify this indistinguishability by the function-hiding property of gFE . For all $i, j \in [2]$, $\langle \mathbf{f}_i^j, \tilde{\mathbf{f}}_i \rangle + h_i^j \tilde{h}_i$ in H_{12} and that in H_{13} are equal (recall that $c_{2,1} = 0$), which implies, for all $j_1, j_2 \in [2]$, $\sum_{i \in [2]} (\langle \mathbf{f}_i^{j_1}, \tilde{\mathbf{f}}_i \rangle + h_i^{j_1} \tilde{h}_i)$ in H_{12} and that in H_{13} are equal. Thus, the indistinguishability of $\{\mathbf{f}, \tilde{\mathbf{f}}, h, \tilde{h}\}$ between H_{12} and H_{13} is implied by the function-hiding property of gFE .

$H_{13} \approx_c H_{14}$. This indistinguishability can be proven similarly to $H_8 \approx_c H_{10}$.

$H_{14} \approx_c H_{15}$. Due to the game condition defined in Definition 2.1, the queries by the adversary satisfy $\sum_{i, \theta \in [2]} c_{i, \theta} (x_i^{1,1} x_\theta^{1,1} - x_i^{1,0} x_\theta^{1,0}) = 0$, which implies, for all $j_1, j_2 \in [2]$, $\sum_{i \in [2]} (\langle \mathbf{f}_i^{j_1}, \tilde{\mathbf{f}}_i \rangle + h_i^{j_1} \tilde{h}_i)$ in H_{14} and that in H_{15} are equal. Thus, the indistinguishability of $\{\mathbf{f}, \tilde{\mathbf{f}}\}$ between H_{14} and H_{15} is implied by the function-hiding property of gFE .

$H_{15} \approx_c G_1$. It is easy to see that this indistinguishability is implied by the partially function-hiding property of pFE , since, for all $i, j, I, J \in [2]$, $\langle \mathbf{b}_i^j, \tilde{\mathbf{b}}_I^J \rangle$ in H_{15} and that in G_1 are equal.

6 Quadratic Multi-input Functional Encryption

We present our quadratic MIFE scheme for $\mathcal{F}_{m,n,X,C}^{\text{MQF}}$. We define the following functions that relate indices in $[n] \times [m]$ with those in $[mn]$:

- location function, $\text{lo} : [n] \times [m] \rightarrow [mn]$, defined as $\text{lo}(x, y) = (x - 1)m + y$;
- location set function, $\text{ls} : [n] \rightarrow 2^{[mn]}$, defined as $\text{ls}(x) = \{\text{lo}(x, 1), \dots, \text{lo}(x, m)\}$;

<p>qSetup(1^λ) \rightarrow qPP, qMSK</p> <p>$\mathbb{G} \leftarrow \mathcal{G}_{\text{BG}}(1^\lambda), \mathbf{A}_1, \dots, \mathbf{A}_n \leftarrow \mathcal{D}_k, \{\mathbf{w}_{i,j}\}_{i,j \in [mn]} \leftarrow \mathbb{Z}_p^{k+1}, \tilde{\mathbf{U}}_1, \dots, \tilde{\mathbf{U}}_{mn} \leftarrow \mathbb{Z}_p^{k \times k}$</p> <p>$\mathbf{u}_1, \dots, \mathbf{u}_{mn} \leftarrow \mathbb{Z}_p^k, \mathbf{V}_1, \dots, \mathbf{V}_{mn} \leftarrow \mathbb{Z}_p^{k \times k}, \tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_{mn} \leftarrow \mathbb{Z}_p^k$</p> <p>pPP, pMSK \leftarrow pSetup(1^λ), iPP, iMSK \leftarrow iSetup(1^λ), gPP, gMSK \leftarrow gSetup(1^λ)</p> <p>qPP := (\mathbb{G}, pPP, iPP, gPP)</p> <p>qMSK := ($\{\mathbf{A}_i\}_{i \in [n]}$, $\{\mathbf{w}_{i,j}\}_{i,j \in [mn]}$, $\{\tilde{\mathbf{U}}_i, \mathbf{u}_i, \mathbf{V}_i, \tilde{\mathbf{v}}_i\}_{i \in [mn]}$, pMSK, iMSK, gMSK).</p> <p>qEnc(qMSK, i, x_i) \rightarrow qCT$_i$</p> <p>$\mathbf{S} \leftarrow \mathbb{Z}_p^{k \times k}, \tilde{\mathbf{s}}, \mathbf{r}, \mathbf{t} \leftarrow \mathbb{Z}_p^k, L \leftarrow \mathbb{Z}_p, \mathbf{l} := \mathbf{e}_{i/n} \otimes (1, L) \in \mathbb{Z}_p^{2n}, \tilde{\mathbf{l}} := \mathbf{e}_{i/n} \otimes (L, -1) \in \mathbb{Z}_p^{2n}$</p> <p>$\mathbf{b}_{\kappa,1} := (x_{i,\kappa}, 0) \in \mathbb{Z}_p^2, \mathbf{b}_{\kappa,2} := (\mathbf{w}_{\text{lo}(i,\kappa)}^\top (\mathbf{I}_{mn} \otimes \mathbf{A}_i \mathbf{S}), \mathbf{u}_{\text{lo}(i,\kappa)}) \in \mathbb{Z}_p^{(mn+1)k}$</p> <p>$\mathbf{b}_{\kappa,3} := \mathbf{t}^\top \mathbf{V}_{\text{lo}(i,\kappa)} \in \mathbb{Z}_p^k, \mathbf{b}_{\kappa,4} = \mathbf{b}_{\kappa,5} := \mathbf{0} \in \mathbb{Z}_p^m, \mathbf{b}_{\kappa,6} := \mathbf{0} \in \mathbb{Z}_p^{km}, \mathbf{b}_\kappa := (\mathbf{b}_{\kappa,1}, \dots, \mathbf{b}_{\kappa,6})$</p> <p>$\tilde{\mathbf{b}}_{\kappa,1} := (x_{i,\kappa}, 0) \in \mathbb{Z}_p^2, \tilde{\mathbf{b}}_{\kappa,2} := (\mathbf{e}_{\text{lo}(i,\kappa)/mn} \otimes \tilde{\mathbf{s}}, \mathbf{r}^\top \tilde{\mathbf{U}}_{\text{lo}(i,\kappa)}) \in \mathbb{Z}_p^{(mn+1)k}$</p> <p>$\tilde{\mathbf{b}}_{\kappa,3} := \tilde{\mathbf{v}}_{\text{lo}(i,\kappa)}^\top \in \mathbb{Z}_p^k, \tilde{\mathbf{b}}_{\kappa,4} = \tilde{\mathbf{b}}_{\kappa,5} := \mathbf{0} \in \mathbb{Z}_p^m, \tilde{\mathbf{b}}_{\kappa,6} := \mathbf{0} \in \mathbb{Z}_p^{km}, \tilde{\mathbf{b}}_\kappa := (\tilde{\mathbf{b}}_{\kappa,1}, \dots, \tilde{\mathbf{b}}_{\kappa,6})$</p> <p>$\mathbf{d}_\tau := (\mathbf{a}_{i,\tau}^\top \mathbf{S}, 0) \in \mathbb{Z}_p^{k+1}, \tilde{\mathbf{d}} := (\tilde{\mathbf{s}}, 0) \in \mathbb{Z}_p^{k+1}$</p> <p>$\mathbf{f}_1 := (\mathbf{r}, \mathbf{t}) \in \mathbb{Z}_p^{2k}, \mathbf{f}_{2,1} = \dots = \mathbf{f}_{2,n} := \mathbf{0} \in \mathbb{Z}_p^{m^2}, \mathbf{f} := (\mathbf{f}_1, \mathbf{f}_{2,1}, \dots, \mathbf{f}_{2,n}), h := 0$</p> <p>pCT$_{\text{lo}(i,\kappa)} \leftarrow$ pEnc(pMSK, ($\mathbf{l}, [\mathbf{b}_\kappa]_1$)), pSK$_{\text{lo}(i,\kappa)} \leftarrow$ pKeyGen(pMSK, ($\tilde{\mathbf{l}}, [\tilde{\mathbf{b}}_\kappa]_2$))</p> <p>iCT$_{i,\tau} \leftarrow$ iEnc(iMSK, $[\mathbf{d}_\tau]_1$), iSK$_i \leftarrow$ iKeyGen(iMSK, $[\tilde{\mathbf{d}}]_2$), gCT$_i \leftarrow$ gEnc(gMSK, $i, ([\mathbf{f}]_1, [h]_2)$)</p> <p>qCT$_i := (\{\mathbf{pCT}_{\text{lo}(i,\kappa)}, \mathbf{pSK}_{\text{lo}(i,\kappa)}\}_{\kappa \in [m]}, \{\mathbf{iCT}_{i,\tau}\}_{\tau \in [k+1]}, \mathbf{iSK}_i, \mathbf{gCT}_i)$.</p> <p>qKeyGen(qMSK, $\mathbf{c} = \{c_{\mu,\nu}\}_{\mu,\nu \in [2]}$) \rightarrow qSK</p> <p>$\tilde{\mathbf{f}}_{i,1} := \left(\sum_{\substack{\mu \in \text{sl}(i) \\ \nu \in [mn]}} c_{\mu,\nu} \tilde{\mathbf{U}}_\mu \mathbf{u}_\nu, \sum_{\substack{\mu \in [mn] \\ \nu \in \text{sl}(i)}} c_{\mu,\nu} \mathbf{V}_\nu \tilde{\mathbf{v}}_\mu \right) \in \mathbb{Z}_p^{2k}, \tilde{\mathbf{f}}_{i,2,1} = \dots = \tilde{\mathbf{f}}_{i,2,n} := \mathbf{0} \in \mathbb{Z}_p^{m^2}$</p> <p>$\tilde{\mathbf{f}}_i := (\tilde{\mathbf{f}}_{i,1}, \tilde{\mathbf{f}}_{i,2,1}, \dots, \tilde{\mathbf{f}}_{i,2,n}), \tilde{h}_i := 0, \boldsymbol{\sigma}_{i,\theta} := \sum_{\substack{\mu \in \text{sl}(i) \\ \nu \in \text{sl}(\theta)}} c_{\mu,\nu} \mathbf{w}_{\mu,\nu} \in \mathbb{Z}_p^{k+1}$</p> <p>gSK \leftarrow gKeyGen(gMSK, $\{\tilde{\mathbf{f}}_i\}_i, \{\tilde{h}_i\}_i$), qSK := ($\mathbf{c}$, gSK, $\{\boldsymbol{\sigma}_{i,\theta}\}_{i,\theta \in [n]}$).</p> <p>qDec(qCT$_1, \dots, \mathbf{qCT}_n$, qSK) $\rightarrow z$</p> <p>$[z_1]_T := \prod_{\mu,\nu \in [mn]} \mathbf{pDec}(\mathbf{pCT}_\nu, \mathbf{pSK}_\mu)^{c_{\mu,\nu}}, [z_{2,i,\theta}]_T := (\mathbf{iDec}(\mathbf{iCT}_{\theta,1}, \mathbf{iSK}_i), \dots, \mathbf{iDec}(\mathbf{iCT}_{\theta,k+1}, \mathbf{iSK}_i))$</p> <p>$[z_3]_T := \mathbf{gDec}(\mathbf{gCT}_1, \dots, \mathbf{gCT}_n, \mathbf{gSK}), [z]_T := [z_1 - \sum_{i,\theta \in [n]} \langle \mathbf{z}_{2,i,\theta}, \boldsymbol{\sigma}_{i,\theta} \rangle - z_3]_T$.</p> <p>Searches for z within the range of $z \leq m^2 n^2 C X^2$</p>

Fig. 22. Our n -input quadratic MIFE scheme.

- slot function, $\text{sl} : [mn] \rightarrow [n]$, defined as $\text{sl}(x) = \lceil x/m \rceil$;
- entry function, $\text{en} : [mn] \rightarrow [m]$, defined as $\text{en}(x) = x - m(\text{sl}(x) - 1)$.

Note that we have $\text{lo}(\text{sl}(x), \text{en}(x)) = x$ for all $x \in [mn]$. Let \mathcal{D}_k be a matrix distribution. Let **pFE** = (**pSetup**, **pEnc**, **pKeyGen**, **pDec**) be an FE scheme for $\mathcal{F}_{2n, 2+(mn+2)k+(2+k)m, \mathbb{G}}^{\text{PIP}}$ (Definition 3.2), **iFE** = (**iSetup**, **iEnc**, **iKeyGen**, **iDec**) be an FE scheme for $\mathcal{F}_{k+1, \mathbb{G}}^{\text{IP}}$ (Definition 3.1), and **gFE** = (**gSetup**, **gEnc**, **gKeyGen**, **gDec**) be an FE scheme for $\mathcal{F}_{2k+m^2n, 1, n, \mathbb{G}}^{\text{MGIP}}$ (Definition 4.2). We construct our quadratic MIFE scheme **qFE** = (**qSetup**, **qEnc**, **qKeyGen**, **qDec**) from **pFE**, **iFE**, and **gFE** as shown in Fig. 22.

Due to space constraints, we present the proof of correctness and security analysis of our scheme in the full version.

References

1. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_19
2. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing Inner-Product Functional Encryption. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 128–157. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_5
3. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
4. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_20
5. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. Cryptology ePrint Archive, Report 2020/577 (2020). <https://eprint.iacr.org/2020/577>
6. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_21
7. Ananth, P., Jain, A., Lin, H., Matt, C., Sahai, A.: Indistinguishability obfuscation without multilinear maps: new paradigms via low degree weak pseudorandomness and security amplification. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 284–332. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_10
8. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15
9. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_3
10. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 470–491. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_20
11. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th FOCS, pp. 171–190. IEEE Computer Society Press, October 2015
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16

13. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_12
14. Chen, J., Wee, H.: Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 277–297. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_16
15. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_24
16. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 164–195. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_7
17. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the k -linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 245–277. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_9
18. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for Diffie-Hellman assumptions. *J. Cryptol.* **30**(1), 242–288 (2017)
19. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, pp. 40–49. IEEE Computer Society Press, October 2013
20. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016, Part II. LNCS, vol. 9563, pp. 480–511. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_18
21. Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 95–120. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_4
22. Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_32
23. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) TCC 2016, Part II. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14
24. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003 (2020). <https://eprint.iacr.org/2020/1003>
25. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
26. Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 544–562. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_29

27. Libert, B., T̃ițiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_18
28. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 599–629. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_20
29. O’Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010). <http://eprint.iacr.org/2010/556>
30. Tomida, J.: Tightly secure inner product functional encryption: multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 459–488. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_16
31. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 408–425. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45871-7_24



Functional Encryption for Turing Machines with Dynamic Bounded Collusion from LWE

Shweta Agrawal¹ (✉), Monosij Maitra², Narasimha Sai Vempati¹, and Shota Yamada³

¹ IIT Madras, Chennai, India

shweta.a@cse.iitm.ac.in

² TU Darmstadt, Darmstadt, Germany

monosij.maitra@tu-darmstadt.de

³ AIST Japan, Tsukuba, Japan

yamada-shota@aist.go.jp

Abstract. The classic work of Gorbunov, Vaikuntanathan and Wee (CRYPTO 2012) and follow-ups provided constructions of bounded collusion Functional Encryption (FE) for circuits from mild assumptions. In this work, we improve the state of affairs for bounded collusion FE in several ways:

1. *New Security Notion.* We introduce the notion of *dynamic* bounded collusion FE, where the declaration of collusion bound is delayed to the time of encryption. This enables the encryptor to dynamically choose the collusion bound for different ciphertexts depending on their individual level of sensitivity. Hence, the ciphertext size grows linearly with its own collusion bound and the public key size is independent of collusion bound. In contrast, all prior constructions have public key and ciphertext size that grow at least linearly with a fixed bound Q .
2. *CPFE for circuits with Dynamic Bounded Collusion.* We provide the first CPFE schemes for circuits enjoying dynamic bounded collusion security. By assuming identity based encryption (IBE), we construct CPFE for circuits of *unbounded* size satisfying *non-adaptive* simulation based security. By strengthening the underlying assumption to IBE with receiver selective opening security, we obtain CPFE for circuits of *bounded* size enjoying *adaptive* simulation based security. Moreover, we show that IBE is a necessary assumption for these primitives. Furthermore, by relying on the Learning With Errors (LWE) assumption, we obtain the first *succinct* CPFE for circuits, i.e. supporting circuits with unbounded size, but fixed output length and depth. This scheme achieves *adaptive* simulation based security.
3. *KPFE for circuits with dynamic bounded collusion.* We provide the first KPFE for circuits of unbounded size, but bounded depth and output length satisfying dynamic bounded collusion security from LWE. Our construction achieves *adaptive* simulation security improving security of [20].
4. *KP and CP FE for TM/NL with dynamic bounded collusion.* We provide the first KPFE and CPFE constructions of bounded collusion functional encryption for Turing machines in the public key setting from LWE. Our constructions achieve non-adaptive simulation based security. Both the input and the machine in our construction can be of *unbounded* polynomial length.

We provide a variant of the above scheme that satisfies *adaptive* security, but at the cost of supporting a smaller class of computation, namely Non-deterministic Logarithmic-space (NL). Since NL contains Nondeterministic Finite Automata (NFA), this result subsumes *all* prior work of bounded collusion FE for uniform models from standard assumptions [7, 9].

1 Introduction

Functional encryption [13, 29] is a generalization of public key encryption which allows fine grained control on disclosure of encrypted data. In functional encryption (FE), a secret key is associated with a function f , a ciphertext is associated with an input x and decryption reveals $f(x)$ and nothing more. Security requires that any collusion of users cannot learn more about the ciphertext beyond what they are individually authorized to learn – this property is known as *collusion resistance*.

In a classic work, Gorbunov, Vaikuntanathan and Wee [21] provided the first construction of Functional Encryption for circuits in the *bounded collusion* model – namely in a model where the scheme may generate an unbounded number of keys but security holds only against an adversary who obtains at most an a-priori bounded number of keys, say Q . Their construction supports all polynomial sized circuits, and is based on the existence of public key encryption (PKE) and pseudorandom generators (PRG) in NC^1 . Since circuits are a powerful model of computation, this work provides a strong feasibility result, and moreover, from weak assumptions. Subsequent work provided other useful improvements: the work of Ananth and Vaikuntanathan [12] removed the assumption of PRGs so that the resulting scheme relies on the minimal assumption of PKE, while a series of works [1, 8, 12] improved the dependence of the public key and ciphertext on the collusion bound Q . A parallel line of work has studied the construction of functional encryption schemes supporting unbounded collusions [2, 10, 17, 23, 24], from standard assumptions, culminating in the recent breakthrough of Jain, Lin and Sahai [24] which achieves this much sought-after goal. However, this intricate construction relies on several assumptions including pairings making it quantum insecure, and has many complex reductions that incur significant loss in efficiency. Hence, it remains meaningful to consider simpler, plausibly post-quantum constructions, even in weaker security models.

Limitations of Prior Work. Despite their success, existing constructions of FE in the bounded collusion model suffer from at least three major drawbacks: i) the collusion bound Q must be declared at setup time and is fixed once and for all, ii) in the public key setting, these constructions support only the circuit model of computation and iii) all constructions that we are aware of are in the *key policy* setting (KPF), where the function f is embedded in the secret key and the input x is embedded in the ciphertext of the FE scheme. The dual, ciphertext policy setting (CPF), where the roles of f and x are swapped, is more natural in several applications but has received much less attention. We discuss each of these limitations in turn.

The first limitation pervades all prior work to the best of our knowledge and is quite a significant drawback in our opinion. Since the collusion bound must be declared at setup time, all data encrypted under the scheme must necessarily be subject to the same level of collusion-resistance. Thus, the practitioner is forced to choose a collusion bound which is strong enough for the *most sensitive* information that may ever be encrypted under the scheme – this implies that Q is an upper bound on collusion resistance. However, not all information has the same level of sensitivity. Consider an organization: messages involving a potential merger with another organization, or a case of harassment which must be investigated are significantly more delicate than routine exchanges.

It is desirable that such sensitive messages are protected even if a large number of key holders collude. On the other hand, for routine ciphertexts decrypted via function keys (such as checking whether some encrypted message is spam or not), such a strong level of collusion resistance is unnecessary. Moreover, the collusion bound Q impacts the size of the public key and ciphertext of the scheme. Thus, if Q is large, then the public key as well as every ciphertext generated by the scheme is forced to grow at least linearly in Q leading to a prohibitive impact on efficiency.

Regarding the second limitation, it is well known that being non-uniform, the circuit model is ill-suited for several applications [5, 9, 11, 19]. In particular, circuits force the size of the input to be fixed a-priori which in turn necessitates instantiation of the scheme with an upper bound on data size. This again leads to loss in efficiency and is ill-suited to datasets of dynamic size. Moreover, circuits incur *worst case* running time over all inputs, which is also clearly undesirable in practice. Finally, all constructions of CPFE for circuits that we are aware of, proceed via the universal circuit route, i.e. consider the universal circuit $U(f, x) = f(x)$ and construct KPFE with the circuit $U(\cdot, x)$ in the secret key and f represented as a string, in the ciphertext. Aside from the loss of efficiency that results by this transformation (now, both the input and the function grow with the maximum circuit size), this transformation restricts the CPFE scheme to *bounded* size circuits. This is dissatisfying, as much in practice as in theory. The ciphertext policy variant of FE is desirable in many applications. Even in the special case of attribute based encryption (ABE), the ciphertext policy model allows an access control policy f to be embedded against a secret message m in the ciphertext. The secret key contains user attributes x which represent the various roles of a user, such as institute, department, date of joining and such others. Decryption succeeds to recover m if and only if the user's attributes satisfy the access control policy in the ciphertext. It is arguably more natural to have an access control policy apply to a secret message than to a user. Another application scenario is when the function f is proprietary and must be hidden via encryption, while the data can be made publicly available. For instance, suppose a government wants to enable citizens to run useful algorithms developed by different research labs on some public data, e.g. census data. The government publishes a secret key for the data x , the labs publish ciphertexts with their respective algorithms f_i (of arbitrary size) and anyone can compute $f_i(x)$. The research labs want to keep their algorithms secret from competitors (but the government agency is trusted) so the function is encrypted. The data is public but becomes available asynchronously and independently of the function(s), so the labs cannot compute the function outputs each time. For example, new census data may be published every year but the same programs can be used every year. Computing private programs on public data is analogous to the setting of obfuscation, except that here the government agency can be trusted, making it simpler than obfuscation. Given the many natural applications of CPFE, it is undesirable to settle for a limited generic transformation via KPFE.

State of the Art. While FE for uniform models of computation has been studied [5, 9, 11, 19], direct constructions from standard assumptions, supporting unbounded length inputs, are few and far-between. In the public key setting, the FE scheme by

Agrawal and Singh [9] supports Turing machines and is based on the Learning With Errors assumption. However, this scheme only supports a single key request by the adversary in the security game. In the symmetric key setting, the recent work of Agrawal, Maitra and Yamada [7] provided a construction of FE for non-deterministic finite automata (NFA) which is secure against bounded collusions of arbitrary size. However, generalizing this construction to the public key setting and to stronger models like Turing machines was left open. To the best of our knowledge, all bounded collusion FE schemes suffer from the fixed collusion bound, and the only CPFE scheme that supports unbounded sized circuits is that by Sahai and Seyalioglu [28], which is only single key secure.

1.1 Our Results

In this work, we improve the state of affairs in several ways:

1. *New Security Notion.* We introduce the notion of *dynamic* bounded collusion FE, where the declaration of collusion bound is delayed to the time of encryption. This enables the encryptor to dynamically choose the collusion bound for different ciphertexts depending on their individual level of sensitivity. Hence, the ciphertext size grows linearly with its own collusion bound and the public key size is independent of collusion bound. In contrast, all prior constructions have public key and ciphertext size that grow at least linearly with a fixed bound Q . All our constructions satisfy our new security notion – we also refer to our new notion as achieving *delayed* collusion resistance.
2. *CPFE for circuits with Dynamic Bounded Collusion.* We provide the first CPFE schemes for circuits enjoying dynamic bounded collusion security. In more detail:
 - By relying on the assumption of identity based encryption (IBE), we construct CPFE for circuits of *unbounded* size, output length and depth satisfying *non adaptive* simulation based security. Recall that non-adaptive simulation security refers to a game where the attacker must make all its key requests before obtaining the challenge ciphertext [13].
 - By strengthening the underlying assumption to IBE with receiver selective opening security, we obtain CPFE for circuits of *bounded* size, output length and depth enjoying *adaptive* simulation based security¹. Moreover, we show that IBE is a necessary assumption for these primitives.
 - By relying on the Learning With Errors (LWE) assumption, we obtain a *succinct* CPFE for circuits – namely, supporting circuits with unbounded size, but fixed output length and depth, which achieves *adaptive* simulation based security.
3. *KPFE for circuits with dynamic bounded collusion.* We provide the first KPFE for circuits of unbounded size, but bounded depth and output length satisfying dynamic bounded collusion. Our construction relies on LWE and achieves adaptive simulation based security. This improves the security of succinct KPFE by Goldwasser et al. [20].

¹ For the knowledgeable reader, the lower bound from [13] does not apply because there is only one challenge ciphertext, with bounded output length in the security game.

4. *KP and CP FE for TM/NL with dynamic bounded collusion.* We provide the first KPFE and CPFE constructions of bounded collusion functional encryption for Turing machines in the public key setting from LWE. Such a result was not known even with fixed collusion resistance to the best of our knowledge. Our constructions achieve non-adaptive simulation based security. In terms of functionality: a secret key in our KPFE construction encodes an TM M , a ciphertext encodes a message x and decryption allows recovery of $M(x)$ and nothing else. Both the input x and the machine M in our construction can be of *unbounded* polynomial length but the ciphertext size grows with the upper bound on the running time of the Turing machine on the given input x . Given RAM access to the ciphertext, the scheme enjoys input specific decryption time.
5. *Adaptive Bounded Collusion FE for NL with dynamic bounded collusion.* The above construction guarantees non-adaptive security while supporting general Turing machines. We also consider a variant of the above scheme that satisfies stronger adaptive security, but at the cost of supporting smaller class of computation Non-deterministic Logarithmic-space (NL). Since NL contains Nondeterministic Finite Automata (NFA), this result subsumes *all* prior work of bounded collusion FE for uniform models from standard assumptions [7, 9].

1.2 Our Techniques

In this section, we provide an overview of our techniques. At a high level, our work addresses two broad challenges: obtaining stronger security guarantees via dynamic collusion and achieving more powerful functionality via general TM or NL. We examine each of these in turn.

Dynamic Bounded Collusion. Observe that the problem of constructing FE with delayed collusion bounds has not been studied until our work, even for bounded size circuits. A first observation is that in such an FE scheme, it is *necessary* that the efficiency of the setup and key generation algorithms are independent of (or dependent only poly-logarithmically on) the collusion bound Q . To the best of our knowledge, previous bounded FE schemes such as [1, 8, 21] do not satisfy this property. However, the recent construction by Ananth and Vaikuntanathan [12] (AV19) does satisfy one half of this requirement – it enjoys a key generation algorithm which is efficient in this sense. Unfortunately, the setup algorithm of their construction runs in time that grows with Q . Our first step will be to remove the dependency on Q from the setup algorithm.

Improving AV19 to remove setup dependence on Q . To begin, we recap some relevant ideas from their construction. Their construction is generic: they construct Q -bounded FE scheme from a single key FE scheme. For concreteness, we instantiate the single key FE scheme with the concrete one by Sahai and Seyalioglu [28]. While their construction is key policy, we adapt it to the ciphertext policy setting in what follows. At a very high level, their Q bounded FE scheme runs Q subsystems in parallel, such that each subsystem in turn runs N instances of the SS10 scheme. Since their construction is optimized using an elegant combinatorial argument, they obtain a secret key size of poly rather than $O(Q \cdot \text{poly})$. This optimized construction forms the starting point of our

work. The specific details of their final construction are not relevant to this overview: we note only some salient features. Their construction makes use of an “MPC style” secret sharing scheme, where an input circuit C is divided into shares $\hat{C}_1, \dots, \hat{C}_N$. Now, n out of N parties, without any interaction, perform some computation on their shares corresponding to input x , to obtain partial outputs $\hat{y}_1, \dots, \hat{y}_n$. These n partial outputs are then combined to obtain output $y = C(x)$.

Using the above secret sharing scheme, the AV19 construction may be summarized as follows. The setup algorithm generates several (the exact number is not relevant for us) public and secret key pairs of a PKE scheme. To encrypt a circuit C , the encryptor computes many shares of C as described above. These shares \hat{C}_i are then hardwired into garbled circuits which, given input x , compute \hat{y}_i using the above method. The labels of these garbled circuits are encrypted using the public keys provided by the setup algorithm. The function key for x is a set of PKE secret keys that depend on x (again, the precise dependence is not required here). The decryptor first uses the PKE secret keys to recover the appropriate labels of the garbled circuit corresponding to x . Then the garbled circuit is evaluated to obtain shares \hat{y}_i of the decryption result. These shares are then combined to obtain $C(x)$.

The reason why their setup algorithm takes time linear in the collusion bound Q is that the number of public keys required by their scheme is linear in Q . Having unrolled their construction when instantiated with [28], our approach to removing this dependency is simple: we replace these public keys of PKE with a single master public key of an identity based encryption (IBE) scheme. Then, encryption with PK_i is replaced by an encryption with $\text{IBE.Enc}(\text{IBE.mpk}, i, \cdot)$, where i is the identity. The intuition for security is the same as the original construction. Thus, we use the power of the IBE to hide the labels of the garbled circuits, use the power of the garbled circuits to hide information other than the decryption shares, and finally use the power of the secret sharing scheme to hide the circuit C .

We show that if we desire adaptive simulation (AD-SIM) security for the resultant construction, we need an IBE satisfying the stronger security notion of *receiver selective opening security* [25]. This is for a reason similar to why [12, 21] required non-committing security for the underlying public key encryption. Since the length of the message that can be encrypted using an IBE with receiver selective opening security is bounded, so is the size of the circuits that can be encrypted in our CPFE scheme. If we relax the security notion and consider non-adaptive (NA-SIM) security, we can use IBE with standard IND-CPA security. This allows us to encrypt a message of unbounded length, which in turn allows us to encrypt circuits with unbounded size. A simple adaptation of the lower bound of Boneh et al. [13] shows that to support unbounded sized circuits, NA-SIM security is optimal².

² Consider a circuit C^* with unbounded size and unbounded output length. Let us say that this circuit has hardwired with random string s of length ℓ , and upon an input x , the circuit ignores the input and outputs s . Here, ℓ is unbounded. Now if the attacker makes even a single key request for some x^* after seeing the challenge ciphertext corresponding to C^* , the simulator is faced with the impossible task of embedding a random string of length ℓ into a fixed sized secret key. Hence, the adversary must not be allowed post-challenge key requests when circuits of unbounded output length are supported.

Supporting Delayed Collusion in Security. So far, we have constructed bounded CPFE schemes whose setup and key generation algorithms run in time independent of Q . However, this is only necessary and not sufficient to construct FE with delayed collusion bound. Once the system is set up with the bound Q , this will still only be secure against a collusion of size Q . Our next step is to remove this restriction so that the encryptor can choose the bound flexibly, and in particular, differently for each ciphertext.

Here, our crucial observation is that the setup and key generation algorithms of the scheme can be run even for super polynomial Q , thanks to their efficiency properties. Hence, we may use the “powers of two trick” [19], where we run the system with different collusion bounds $Q = 2, 2^2, \dots, 2^\lambda$. The setup and key generation algorithms are run for these λ subsystems in parallel. When we encrypt the message, the encryptor chooses the smallest 2^i that exceeds the bound Q it wants and encrypts the message using the i -th subsystem. Decryption is performed using the secret key for the i -th instance of the subsystem. The resulting scheme inherits the efficiency and security properties of the subsystem. Namely, if the subscheme is NA-SIM (respectively AD-SIM) secure and supports unbounded (respectively bounded) circuits, so does the resulting scheme. Thus, we obtain two schemes with incomparable properties. Please see Sect. 3 for details.

Observe that the construction of bounded collusion FE in AV19 can be based on the minimal assumption of plain PKE [12]. On the other hand, our constructions described above rely on the stronger primitive of IBE. It is natural to ask whether we can base the security of the construction to weaker primitives such as PKE. We answer this question negatively. In our full version [6] we argue that the usage of IBE is unavoidable, by showing that FE with dynamic bounded collusion for very small class of functionalities already implies IBE.

Supporting More General Function Classes. Next, we describe our techniques for supporting more flexible models of computation, namely Turing machines or NL. The main difficulty of constructing FE for these function classes is to handle unbounded length inputs and unbounded size machines simultaneously. To address this, we borrow a trick from the work of Agrawal, Maitra and Yamada [7]: instead of trying to handling them at once, we construct intermediate schemes that can handle an unbounded size object on one side, but bounded size object on the other. Towards this, we construct KPFE and CPFE schemes with dynamic bounded collusion that support unbounded size circuits, but with bounded output length and depth (note that input length is always fixed). Later, we will see how to compile these to construct FE for more general function classes. Note that in the previous step we already constructed a CPFE scheme that can handle circuits with unbounded size even without restrictions on depth and output length. However, this construction was only NA-SIM secure. Here, we aim to construct schemes with AD-SIM security.

Succinct KPFE and CPFE. Let us start with the construction of KPFE that can support unbounded size circuits. Note that such FE schemes have already been constructed by the previous works under the name of succinct FE [1, 20]. However, they do not satisfy the security requirement that we want. Namely, they are (conventional) bounded collusion schemes and do not satisfy the delayed collusion property. In addition, they only satisfy NA-SIM security. Here, we upgrade the security of existing succinct FE

schemes so that they satisfy the delayed collusion property and AD-SIM security with the help of our CPFE scheme for *bounded* circuits that already satisfies the desired security properties. In more detail, we combine succinct single-key KPFE, denoted by 1KPFE with our AD-SIM secure CPFE for bounded circuits, to obtain a new succinct KPFE with the desired security properties.

At a high level, the construction works as follows. The master public key and master secret key of the final KPFE scheme are those of the CPFE. To encrypt a message x for a collusion bound 1^Q , the encryptor first constructs a circuit $1\text{KPFE.Enc}(\cdot, x)$ ³, which is an encryption algorithm of the single-key KPFE that takes as input a master public key of the single-key KPFE and outputs an encryption of the message x under the key. The encryptor then encrypts the circuit using the CPFE scheme with respect to the bound 1^Q . To generate a secret key for a circuit C , we first freshly generate a master key pair of the single-key KPFE (1KPFE.mpk , 1KPFE.msk). We then generate a CPFE secret key CPFE.sk corresponding to the string 1KPFE.mpk and then generate secret key 1KPFE.sk_C for the circuit C of the single-key KPFE scheme. The final secret key is $(\text{CPFE.sk}, 1\text{KPFE.sk}_C)$. Decryption is done by first decrypting the CPFE ciphertext using the CPFE secret key to recover $1\text{KPFE.Enc}(1\text{KPFE.mpk}, x)$ and then decrypting it using the secret key 1KPFE.sk_C of the single-key KPFE scheme to recover $C(x)$.

We discuss the efficiency of the above scheme. First we claim that the above scheme is succinct (or equivalently, can deal with unbounded size of circuits). This is the case even though underlying CPFE can only deal with bounded size circuits, since the size of circuits that should be supported by the encryption algorithm of CPFE is $|1\text{KPFE.Enc}(\cdot, x)| = \text{poly}(\lambda, |x|)$, which is independent of the size of circuits by the succinctness of the underlying 1KPFE scheme. Next, we discuss the security of the scheme. Intuitively speaking, by the security of the underlying CPFE, the adversary can obtain no information beyond the decryption result of the CPFE. This means that, the adversary only obtains the information $\{1\text{KPFE.Enc}(1\text{KPFE.mpk}^{(i)}, x), 1\text{KPFE.sk}_{C^{(i)}}^{(i)}\}_{i \in [Q]}$ for Q freshly generated, independent instances. In turn, this implies that the adversary can only obtain the information of $\{C^{(i)}(x)\}_{i \in [Q]}$ by the single-key security of the underlying KPFE, as desired. A formal argument shows that the resulting KPFE scheme inherits AD-SIM security of the CPFE, even if the underlying single-key KPFE is only NA-SIM secure. We refer to the Sect. 4 for details.

Next, we discuss the adaptation to the CPFE setting. In this construction, we use a reusable garbled circuit scheme [20] instead of single-key succinct KPFE. Recall that a reusable garbled circuit is a symmetric key variant of single key succinct KPFE, where the circuit in the secret key is also hidden. Now, to encrypt a circuit C in our CPFE, we run the garbling algorithm of the reusable garbled circuit scheme on input C to obtain the garbled version of C as well as some secret information for input garbling. We then construct an input encoding circuit that takes as input x and outputs an encoded version of it using the secret information generated above. Then, we encrypt this input encoding circuit using the underlying CPFE. The final ciphertext consists of the garbled circuit and the encrypted circuit. To generate a secret key for x , we use the key generation

³ The description here is oversimplified – in fact we need a PRF to derive the randomness for the encryption.

algorithm of the underlying CPFE scheme to obtain a secret key for x . Decryption requires running the decryption algorithm of the underlying CPFE scheme to obtain the encoded version of input x and then using this result with the garbled version of the circuit to recover $C(x)$. The resulting scheme is AD-SIM secure and supports unbounded size circuits. Please see our full version [6] for details.

Handling Unbounded Inputs. So far, we have constructed KPFE and CPFE schemes that can handle unbounded size circuits but with fixed input size. However, for obtaining FE for Turing machines, we have to be able to encrypt unbounded length inputs and this is clearly insufficient. To enable this, we use the “delayed encryption” technique by Goyal, Koppula, and Waters (GKW16) [22]. Intuitively, the technique uses the power of garbled circuits and IBE to transport us to a world where there are infinitely many instances of FE $\{\text{mpk}_i\}_{i \in \mathbb{N}}$ – the encryptor can choose a master public key for some index j and encrypt the message. A secret key is associated with some index k and the decryption is possible iff $j = k$.

In more detail, the GKW16 scheme works as follows. During setup, the master public key and master secret key of an IBE scheme are generated. To encrypt a message x using the j -th instance of the FE scheme as described above without knowing the corresponding mpk_j , the encryptor first constructs a circuit $\text{Enc}(\cdot, x)$ that takes as input the master public key mpk_j for the j -th instance and outputs the ciphertext $\text{Enc}(\text{mpk}_j, x)$. The encryptor then garbles this encryption circuit to obtain the corresponding garbled circuit and set of labels. Then, the encryptor encrypts each pair of labels with the IBE, where the identity for which a label is encrypted encodes the index j , the position of the label and a single bit. Note that the above step can be done without knowing mpk_j . Correspondingly, the key generation algorithm computes IBE secret keys for the correct bits of mpk_k , which together with the IBE ciphertexts allow the decryptor to recover the labels corresponding to mpk_j in the j^{th} garbled circuit when $j = k$. This lets the decryptor evaluate the garbled circuit to retrieve the ciphertext $\text{Enc}(\text{mpk}_j, x)$ as desired.

With this technique, we make progress towards our goal, because we can encrypt a message of any length by the scheme, rather than only being able to encrypt a message of fixed length. However, this is still not enough, since the decryption is possible only when the index j and k match. For example, let us imagine that we want to construct FE for Turing machine using infinitely many instances of FE for circuits, where the i -th instance of FE supports circuits with input length i . Let t be an upper bound on the runtime of the TM on input x . If we encrypt a message $(x, 1^t)$ as an input to a Turing machine, we may encrypt it using $|x, 1^t|$ -th instance of the FE. On the other hand, to generate a secret key for a Turing machine M , we convert the machine into a circuit $C_M(\cdot)$ and generate a secret key for it. However, it is unclear how to define the input length of the circuit. If the input length does not match $|x, 1^t|$, the decryption is impossible. Meanwhile, the entity who generates the secret key does not know the input length $|x, 1^t|$, so is stuck.

To resolve the above problem, we incorporate a trick by Agrawal, Maitra and Yamada [7] used to support unbounded inputs for an NFA machine in the context of ABE. They construct two restricted ABE schemes for NFA: one that supports decryption in the case where the length $|x|$ of the input x is larger than the size $|M|$ of the machine M and one that supports the case where $|x| \leq |M|$. Then, they run the restricted schemes in parallel. In the decryption algorithm, these sub-schemes complement each other. Namely, we use the first sub-scheme to decrypt a ciphertext if

$|x| > |M|$ and the second otherwise. Though they introduce the trick in the context of ABE, this perfectly works in the context of FE as well. A hurdle is that their technique works only in the secret key setting, since the encryptor is required to know a master secret in order to generate unbounded instances of FE (proportional to its input length) on the fly. However, we show that in conjunction with the technique from [22] described above, this idea can be made to work in the public key setting as well. In a nutshell, this technique lets us encode x and M in multiple slots of the FE instances so that they always intersect, instead of encoding them on a single slot like in [22].

Onward to FE for TM. Armed with these techniques, let us try constructing FE for TM. As discussed above, our construction handles the cases $|(x, 1^t)| \leq |M|$ and $|(x, 1^t)| > |M|$ separately. Let us begin with the former using our KPFE that supports unbounded size circuits.

By the technique of [22], we can assume that we are in a world where there are infinitely many KPFE instances available and the i -th instance supports circuits with input length i . To encrypt a message x with respect to the time bound 1^t , we use the $|(x, 1^t)|$ -th instance of the KPFE. To generate the secret key for a Turing machine M on the other hand, we encode M into a set of circuits $C_{i,M}$ for $i = 1, \dots, |M|$, where $C_{i,M}$ is a circuit that takes as input a string $(x, 1^t)$ and then run the machine M for t steps and outputs the result. We then generate secret keys for $C_{i,M}$ using the i -th instance of KPFE for all of $i \in [|M|]$. This is possible even for unbounded M , because each KPFE instance supports unbounded size circuits. The decryption is possible when $|(x, 1^t)| \leq |M|$ by using the $|(x, 1^t)|$ -th instance. However, it is evident that this is an incomplete scheme, since the decryption is not possible when $|(x, 1^t)| > |M|$.

To complement this, we next construct a scheme that deals with the case of $|(x, 1^t)| > |M|$ using our unbounded CPFE scheme. To encrypt a message $(x, 1^t)$ we convert it into a circuit $\{U_{i,x,t}\}_{i \in [|x, 1^t|]}$, where $U_{i,x,t}$ is a circuit that takes as input a string M of length i , interprets it as a description of a Turing machine, and then runs it on input x for t steps to obtain the result. We then encrypt the circuit $U_{i,x,t}$ using the i -th instance of the FE for all of $i \in [|x, 1^t|]$. This requires the underlying CPFE scheme to support unbounded size of circuits. Since our NA-SIM secure CPFE construction supports such circuits, we can use this here. On the other hand, our CPFE scheme with AD-SIM security cannot be used here, because the scheme can only support circuits with bounded depth, which prohibits us from running the Turing machine inside the circuit for t steps, where t may be arbitrarily large. To generate a secret key for a Turing machine M , we use the $|M|$ -th instance of the FE. It can be seen that the decryption is possible in this scheme when $|(x, 1^t)| > |M|$. Having the construction for the cases of $|(x, 1^t)| > |M|$ and $|(x, 1^t)| \leq |M|$, we can obtain the final construction by running them in parallel. The final scheme is NA-SIM secure, because the underlying CPFE scheme is NA-SIM secure (even though the KPFE scheme satisfies the stronger AD-SIM security). Please see Sect. 5 for details.

Above, the ciphertext size grows with the t , the upper bound on the runtime of the TM on a given input x . We emphasize that t is not a global bound but can vary with each input x , and is therefore unbounded. While we do not know how to remove this dependence using our current techniques, we remark that decryption time can still be input specific using the “powers of two” trick from Goldwasser et al. [19]. This requires the

decryption algorithm to have RAM access to the ciphertext. In more detail, the encryptor may repeat the encryption procedure for $\lceil \log_2 t \rceil$ possible values of TM runtime, with values 2^i for $i \in [\lceil \log_2 t \rceil]$. The decryptor can start with the ciphertext corresponding to the smallest value and proceed to the next ciphertext only if the previous decryption did not result in a valid output⁴. This ensures that the scheme enjoys input specific decryption time.

FE for NL with Adaptive Security. The above construction guarantees NA-SIM security while supporting general Turing machines. We also consider a variant of the above scheme that satisfies stronger AD-SIM security, at the cost of supporting smaller class of computation NL. This is achieved by using AD-SIM secure FE for both the building blocks of KPFE and CPFE. Recall that the reason why we cannot use AD-SIM secure CPFE in the above construction was that it was not possible to run the Turing machine for an unbounded number of steps inside a circuit of fixed depth. This issue arises because the Turing machine is run sequentially. In a nutshell, our next idea is to parallelize computation so that the depth for the corresponding circuit can be bounded by some fixed polynomial. Such a parallelization of the computation is not known to be possible for general Turing machines, but we can do it for NL, which is more restrictive. To do so, we represent the computation of NL as a multiplication of matrices as was done in [26]. First, we enumerate all the possible internal configurations of the Turing machine M on input x that may appear during the computation. The number of such internal configurations can be bounded by some polynomial, since the length of the working tape is logarithmic. We then construct the transition matrix M for the configurations. Then, one can determine whether M accepts the input x within time t by computing M^t due to the properties of the transition matrix. Since the matrix exponentiation can be done by $O(\log t)$ multiplications of the matrix, this can be performed by fixed polynomial depth even for unbounded t (assuming $t < 2^\lambda$). We refer to the full version [6] for details.

1.3 Concurrent Work

A concurrent work [16] independently introduced the notion of dynamic collusion bound for KPFE schemes, which is the same as what we consider in this paper. They obtain simulation secure KPFE schemes for circuits with dynamic collusion resistance. Further, their techniques also significantly overlap with our CPFE constructions in Sect. 3. In particular, they compile existing bounded collusion KPFE schemes [12, 21] with IND-CPA secure IBE to obtain KPFE for circuits satisfying dynamic collusion bound property. However, we also extend our results further to obtain succinct CP/KPFE schemes for circuits with dynamic collusion property, and also to support Turing machines and NL with various security tradeoffs as explained in Sect. 1.2. All these constructions support dynamic collusion resistance. Furthermore, we also argue about the necessity of IBE to achieve dynamic collusion bound property for FE schemes, which is left as an open problem in [16].

⁴ The definition of TM can be easily modified to output “unfinished” if the computation did not conclude in a given number of steps.

2 Preliminaries

In this section, we define some notation and preliminaries that we require. Please see the full version [6] for some additional preliminaries.

2.1 Functional Encryption

Functional encryption (FE) [13, 27, 29] has been traditionally defined in a setting where a trusted key generator holding a master secret key provides authorized users with secret keys corresponding to functions. Such a key, when used to decrypt ciphertexts, reveals only the function of the plaintexts and nothing else. In this subsection, we define the notion of functional encryption (FE) more generally so that it captures the above notion as well as other types of functionalities as special cases.

2.1.1 Syntax and Correctness

Let $R : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}^*$ be a two-input function where \mathcal{X} and \mathcal{Y} denote “message space” and “key attribute space”, respectively. Ideally, we would like to have an FE scheme that handles the relation R directly, where we can encrypt any message $x \in \mathcal{X}$ and can generate a secret key for any key attribute $y \in \mathcal{Y}$. However, in many cases, we are only able to construct a scheme that poses restrictions on the message space and key attribute space. To capture such restrictions, we introduce a parameter prm and consider subsets of the domains $\mathcal{X}_{\text{prm}} \subseteq \mathcal{X}$ and $\mathcal{Y}_{\text{prm}} \subseteq \mathcal{Y}$ specified by it and the function R_{prm} defined by restricting the function R on $\mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}}$. An FE (FE) scheme for $\{R_{\text{prm}} : \mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}} \rightarrow \{0, 1\}^*\}_{\text{prm}}$ is defined by the following PPT algorithms:

$\text{Setup}(1^\lambda, \text{prm}) \rightarrow (\text{mpk}, \text{msk})$: The setup algorithm takes as input the unary representation of the security parameter λ and a parameter prm that restricts the domain and range of the function and outputs the master public key mpk and a master secret key msk .

$\text{Encrypt}(\text{mpk}, x) \rightarrow \text{ct}$: The encryption algorithm takes as input a master public key mpk and a message $x \in \mathcal{X}_{\text{prm}}$. It outputs a ciphertext ct .

$\text{KeyGen}(\text{msk}, y) \rightarrow \text{sk}$: The key generation algorithm takes as input the master secret key msk , and a key attribute $y \in \mathcal{Y}_{\text{prm}}$. It outputs a secret key sk . We assume that y is included in sk .

$\text{Dec}(\text{ct}, \text{sk}) \rightarrow m$ or \perp : The decryption algorithm takes as input a ciphertext ct and a secret key sk . It outputs the message m or \perp which represents that the ciphertext is not in a valid form.

Remark 1 (Bounded collusion variants). In this paper, we mainly focus on FE with bounded collusion security, where the security is guaranteed only when the number of secret keys that the adversary obtains during the security game is below collusion bound Q . To do so, we have to slightly change the syntax above. We consider two different types of syntax for FE depending on when Q is declared.

- The first notion we consider is *bounded collusion FE* [12, 21], where Q is fixed when the system is setup. In more details, we change the syntax of FE defined above so that all the algorithms (Setup, KeyGen, Enc, Dec) take 1^Q as additional input.

- The second notion we consider is a new notion that we call *dynamic bounded collusion FE*. In this notion, the bound Q is specified by the encryptor, not by the setup. Namely, we change the syntax of FE above so that only the encryption algorithms Enc takes 1^Q as input, whereas other algorithms (Setup, KeyGen, Dec) do not.

We note that the requirement that the algorithms run in polynomial time along with the fact that all algorithms in bounded collusion FE take 1^Q as additional input imply that all the algorithms run in polynomial in Q . In the case of FE with dynamic bounded collusion, similar implication holds for the encryption algorithm. However, the running time of Setup and KeyGen should be dependent only on λ and $|\text{prm}|$, whereas the running time of Dec may indirectly depend on Q if the size of the input ciphertext is dependent on Q .

Definition 1 (Correctness). *An FE scheme $\text{FE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ is correct if for all prm , $x \in \mathcal{X}_{\text{prm}}$, and $y \in \mathcal{Y}_{\text{prm}}$,*

$$\Pr \left[\begin{array}{l} (\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \text{prm}) : \\ \text{Dec}(\text{Enc}(\text{mpk}, x), \text{KeyGen}(\text{msk}, y)) \neq R(x, y) \end{array} \right] = \text{negl}(\lambda)$$

where probability is taken over the random coins of Setup, KeyGen and Enc.

2.1.2 Security Notions

As security notions for FE, we define simulation-based notions. We are mainly interested in the bounded collusion settings because the security notion without the collusion bound is shown to be impossible [4]. We first provide the description of the security game for FE with dynamic bounded collusion and then for bounded collusion FE.

Definition 2 (AD-SIM and NA-SIM Security for FE with Dynamic Bounded Collusion). *Let $\text{FE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a (public key) FE scheme with dynamic bounded collusion for the function family $\{R_{\text{prm}} : \mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}} \rightarrow \{0, 1\}^*\}_{\text{prm}}$. For every stateful PPT adversary A and a stateful PPT simulator $\text{Sim} = (\text{SimEnc}, \text{SimKG})$ consider the following experiments:*

$\text{Exp}_{\text{FE}, A}^{\text{real}}(1^\lambda)$:	$\text{Exp}_{\text{FE}, \text{Sim}}^{\text{ideal}}(1^\lambda)$:
1: $\text{prm} \leftarrow A(1^\lambda)$ 2: $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \text{prm})$ 3: $(x, 1^Q) \leftarrow A^{\text{KeyGen}(\text{msk}, \cdot)}(\text{mpk})$	1: $\text{prm} \leftarrow A(1^\lambda)$ 2: $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \text{prm})$ 3: $(x, 1^Q) \leftarrow A^{\text{KeyGen}(\text{msk}, \cdot)}(\text{mpk})$ – Let $(y^{(1)}, \dots, y^{(Q_1)})$ be A 's oracle queries. – Let $\text{sk}^{(a)}$ be the oracle reply to $y^{(a)}$. – Let $\mathcal{V} := \left\{ \left(z^{(a)} := R(x, y^{(a)}), y^{(a)}, \text{sk}^{(a)} \right) \right\}_{a \in [Q_1]}$.
4: $\text{ct} \leftarrow \text{Enc}(\text{mpk}, x, 1^Q)$ 5: $b \leftarrow A^{\mathcal{O}(\text{msk}, \cdot)}(\text{mpk}, \text{ct})$ 6: Output b	4: $(\text{ct}, \text{st}) \leftarrow \text{SimEnc}(\text{mpk}, \mathcal{V}, 1^{ \mathcal{V} }, 1^Q)$ 5: $b \leftarrow A^{\mathcal{O}'(\text{st}, \text{msk}, \cdot)}(\text{mpk}, \text{ct})$ 6: Output b

We emphasize that the adversary A is stateful, even though we do not explicitly include the internal state of it into the output above for the simplicity of the notation. On the other hand, the above explicitly denotes the internal state of the simulator Sim by st . We distinguish between two cases of the above experiment:

1. The adaptive case, where:

- The oracle $\mathcal{O}(\text{msk}, \cdot) = \text{KeyGen}(\text{msk}, \cdot)$ with $1 \leq Q_1 < Q$, and
 - The oracle $\mathcal{O}'(\text{st}, \text{msk}, \cdot)$ takes as input the q -th key query $y^{(q)}$ for $q \in [Q_1 + 1, Q_1 + Q_2]$ and returns $\text{SimKG}(\text{st}, \text{msk}, R(x, y^{(q)}), y^{(q)})$, where $Q_1 + Q_2 \leq Q$
- The FE scheme FE is then said to be simulation secure for one message against adaptive adversaries (AD-SIM-secure, for short) if there is a PPT simulator Sim such that for every PPT adversary A , the following holds:

$$\left| \Pr[\text{Exp}_{\text{FE}, A}^{\text{real}}(1^\lambda) = 1] - \Pr[\text{Exp}_{\text{FE}, \text{Sim}}^{\text{ideal}}(1^\lambda) = 1] \right| = \text{negl}(\lambda) \quad (2.1)$$

2. The non-adaptive case, where $Q_1 \leq Q$ and the oracles $\mathcal{O}(\text{msk}, \cdot)$ and $\mathcal{O}'(\text{msk}, \cdot)$ are both the “empty” oracles that return nothing: The FE scheme FE is then said to be simulation secure for one message against non-adaptive queries (NA-SIM-secure, for short) if there is PPT simulator $\text{Sim} = (\text{SimEnc}, \perp)$ such that for every PPT adversaries A , Eq. (2.1) holds. Note that in the non-adaptive case, we can ignore st since SimKG is not present in the above game and it is never used by other algorithm.

Definition 3 (AD-SIM and NA-SIM Security for bounded collusion FE [21]). Let $\text{FE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a (public key) bounded collusion FE scheme for the function family $\{R_{\text{prm}} : \mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}} \rightarrow \{0, 1\}^*\}_{\text{prm}}$. We define AD-SIM and NA-SIM security for FE by considering the same game as Definition 2 with the following changes:

- We change A to output 1^Q in addition to prm at the beginning of the game.
- All the algorithms run in the experiment $(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{SimEnc}, \text{SimKG})$ take 1^Q as additional input.

We also define weaker notion of the security where the adversary is restricted to always choose $Q = 1$ in the non-adaptive case. If the scheme only satisfies this weaker security notion, we say the scheme is 1-NA-SIM secure.

Remark 2. The above definition allows to argue security of a single instance of FE. For the security proof of our constructions in Sect. 5.1, it is convenient to consider multi-instance version of the above notion. In the multi-instance security variant, the adversary declares the number of instances M at the beginning of the game and interact with each instance as above. In the real world, the adversary interacts with real algorithms in all instances, while in the ideal world, it interacts with simulators in all instances. The adversary can make queries in arbitrary order and make them arbitrarily correlated as long as it respects the restriction for each instance. This multi-instance security notion is easily shown to be equivalent to the single-instance security notion above by simple hybrid argument.

2.1.3 Special Classes of FE

We then define various kinds of FE by specifying the relation.

KPFE for circuits. To define KPFE for circuits, we set $\mathcal{X} = \{0, 1\}^*$ and \mathcal{Y} as the set of all circuits and define $R(x, C) = C(x)$ if the length of the string x and the input length of C match and otherwise $R(x, C) = \perp$. In this paper, we will consider the circuit class $\mathcal{C}_{\text{inp,dep,out}}$ that consists of circuits with input length $\text{inp} := \text{inp}(\lambda)$, depth $\text{dep} := \text{dep}(\lambda)$, and output length $\text{out} := \text{out}(\lambda)$. To do so, we set $\text{prm} = (1^{\text{inp}}, 1^{\text{dep}}, 1^{\text{out}})$, $\mathcal{X}_{\text{prm}} = \{0, 1\}^{\text{inp}}$, and $\mathcal{Y}_{\text{prm}} = \mathcal{C}_{\text{inp,dep,out}}$.

CPFE for circuits. To define CPFE for circuits, we set \mathcal{X} to be the set of all circuits and $\mathcal{Y} = \{0, 1\}^*$ and define $R(C, x) = C(x)$ if the length of the string x and the input length of C match and otherwise $R(x, C) = \perp$. In this paper, we will consider the circuit class \mathcal{C}_{inp} that consists of circuits with input length $\text{inp} := \text{inp}(\lambda)$. To do so, we set $\text{prm} = 1^{\text{inp}}$, $\mathcal{X}_{\text{prm}} = \mathcal{C}_{\text{inp}}$, and $\mathcal{Y}_{\text{prm}} = \{0, 1\}^{\text{inp}}$.

Remark 3. In the definition of KPFE for circuits, even though the input length, output length, and depth of the circuits in $\mathcal{C}_{\text{inp,dep,out}}$ are bounded, the size of the circuits is unbounded. Similar comments hold true for \mathcal{C}_{inp} in the definition of CPFE.

Remark 4. Note that our definition of the KPFE requires that the running time of the encryption algorithm is bounded by $\text{poly}(\lambda, |\text{prm}|) = \text{poly}(\lambda, \text{inp}, \text{dep}, \text{out})$. In particular, the running time should be independent from the size of the circuit being supported by the scheme, which is unbounded. This property is called succinctness in [20].

FE for Turing Machines. To define FE for Turing machines, we set $\mathcal{X} = \{0, 1\}^*$, \mathcal{Y} to be set of all Turing machine, and define $R : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \cup \{\perp\}$ as

$$R((x, 1^t), M) = \begin{cases} 1 & \text{if } M \text{ accepts } x \text{ in } t \text{ steps} \\ 0 & \text{otherwise.} \end{cases}$$

FE for NL. To define FE for NL, we set $\mathcal{X} = \{0, 1\}^*$, \mathcal{Y} to be set of all non-deterministic Turing machines with two tapes, one of which encodes the input and can only be read, whereas the other tape can be read as well as written. When we measure the space complexity of the computation, we consider the space being used for the latter tape. We define $R : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \cup \{\perp\}$ as

$$R((x, 1^t, 1^{2^s}), M) = \begin{cases} 1 & \text{if } M \text{ accepts } x \text{ within } t \text{ steps and spaces} \\ 0 & \text{otherwise.} \end{cases}$$

Note that here, s is in the exponent to reflect the idea that the space for the computation is logarithmically bounded.

We recall the following result by Goldwasser et al. [20] that we will use later in Sect. 4.

Theorem 1 ([20]). *There exists a KPFE scheme KPFE for the circuit class $\mathcal{C}_{\text{inp,dep,out}}$ with 1-NA-SIM security assuming sub-exponential hardness of the LWE problem.*

Remark 5. Note that [20] defines their security notion as *full-simulation* based security. However, as stated in [20] only, their *full-simulation* security is equivalent to the *non-adaptive simulation* security definition from [21], where no post-challenge key queries are allowed. The 1-NA-SIM security given in Definition 3 is implied by the same that is adopted for this work. Thus, [20] is 1-NA-SIM secure according to our Definition 3.

3 CPFE with Dynamic Bounded Collusion

In this section, we construct public-key, ciphertext-policy functional encryption (CPFE) schemes with delayed collusion bound. The first scheme supports *unbounded* polynomial-size circuits and achieves NA-SIM security. The second scheme only supports bounded polynomial-size circuits, but achieves stronger AD-SIM security. Both schemes are obtained by first constructing a bounded FE with special efficiency property (Sects. 3.2 and 3.3) and then converting it into a scheme with delayed collusion bound (Sect. 3.4).

3.1 Preparations

Here, we define reusable, dynamic multi-party computation (RDMPC) protocol in the client-server framework, which is introduced by Ananth and Vaikuntanathan [12] as a useful notion for the construction of bounded FE. The formal definition of the protocol as a tuple of algorithms with its correctness and security definitions appear in our full version [6]. We adapt the syntax and definitions of [12] to our setting.

We provide some intuition on how these algorithms may be used as a multi-party computation in a client-server framework. Similar to [12], here also the setting consists of a single client and N servers. In particular, the client wants to offload an *a priori bounded* number of computations R to these N servers. Each computation is termed as a session. To this end, the protocol consists of two phases as described below:

- An *offline* phase, where the client takes the session count R and a *secret* circuit $C \in \mathcal{C}_{\text{inp}}$ as input, and encodes it (via CktEnc algorithm) as $(\widehat{C}_1, \dots, \widehat{C}_N)$. For all $k \in [N]$, it then sends the k th encoding \widehat{C}_k to the k th server.
- An *online* phase, that is performed for R sessions. The client wishes to delegate an input $x^{(j)}$ in the j th session for some $j \in [R]$. For this, it encodes $x^{(j)}$ (via InpEnc algorithm) as $\widehat{x}^{(j)}$ and forwards it to all the N servers. Hereon, some n out of N servers (formalized via any subset $S \subseteq [N]$ of size n) may come to do some *local* computation (via the Local algorithm) on their own inputs to get a *partial* output encoding. In particular, the u th server may compute the partial output encoding $\widehat{y}_u^{(j)} = \text{Local}(\widehat{C}_u, \widehat{x}^{(j)})$. The final output $C(x^{(j)})$ is obtained by combining these partial output encodings $\{\widehat{y}_u^{(j)}\}_{u \in S}$ via the public evaluation algorithm Decode.

Crucially, the R input encodings $\{\widehat{x}^{(j)}\}_{j \in [R]}$ are generated with randomness *independent* from that of the offline phase. Further the terms “*reusable*” and “*dynamic*” stems from the respective requirements that the secret circuit encoding must be reusable across all R sessions and the final output may be recovered dynamically by *any* subset S of the N servers in each session. We then recall the result from [12] adapted to our setting.

Theorem 2 (Adapted from [12]). *Assuming the existence of one-way functions, there exists an RDMPC protocol with parameter $N = \Theta(R^2\lambda)$, $t = \Theta(R\lambda)$, and $n = \Theta(t)$ for \mathcal{C}_{inp} with any $\text{inp} = \text{poly}(\lambda)$.*

3.2 Basic Construction

Here, we give a construction of bounded CPFE. The construction supports unbounded size circuits and is secure against bounded collusion. A nice feature of the construction is that the running times of the setup and key generation algorithms are independent from the collusion bound. Looking ahead, we leverage this property to upgrade the construction to be an CPFE scheme with *dynamic bounded collusion property* later in Sect. 3.4.

In more details, we provide the description of CPFE for the circuit class \mathcal{C}_ℓ for arbitrary $\ell = \ell(\lambda)$, where \mathcal{C}_ℓ is the set of all polynomial size circuits with input length ℓ . Formally, our FE is for the relation $R_{\text{prm}} : \mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}} \rightarrow \{0, 1\}^*$ where $\text{prm} = 1^\ell$, $\mathcal{X}_{\text{prm}} := \mathcal{C}_\ell$, and $\mathcal{Y}_{\text{prm}} := \{0, 1\}^\ell$ and $R_{\text{prm}}(C, x) = C(x)$, for all $C \in \mathcal{C}_\ell$ and $x \in \{0, 1\}^\ell$.

Ingredients. We now describe the underlying building blocks used to obtain our CPFE construction:

1. A reusable, dynamic MPC protocol for \mathcal{C}_ℓ denoted by $\text{RDMPC} = (\text{CktEnc}, \text{InpEnc}, \text{Local}, \text{Decode})$ with $N = \Theta(R^2\lambda)$, $t = \Theta(R\lambda)$, and $n = \Theta(t)$. We can instantiate this by the protocol by Ananth and Vaikuntanathan [12], which can be based on any one-way function (See Theorem 2). Looking ahead, we will set $R = \lambda$ in our construction and therefore ignore the dependence on R when considering the size of the parameters in the following. We denote the length of an encoding \hat{x} of $x \in \{0, 1\}^\ell$ by $\hat{\ell} = \text{poly}(\lambda, \ell)$. We also denote the size of the circuit $\text{Local}(\hat{C}_j, \cdot)$ by $\hat{s}(\lambda, R, |C|)$, where \hat{C}_j is an output of CktEnc on input a circuit C . By the efficiency properties of RDMPC, we have $\hat{s}(\lambda, |C|) = \text{poly}(\lambda, |\hat{C}_j|) = \text{poly}(\lambda, |C|)$.
2. A garbled circuit scheme $\text{GC} = (\text{GC.Garble}, \text{GC.Eval})$ for circuit class $\mathcal{C}_{\hat{\ell}}$. We can instantiate this by Yao's scheme [30], which can be based on any one-way function. We assume that a label is represented by a binary string. The length of a label depends on the size of circuits that are garbled. We denote the length of the labels obtained by garbling a circuit of size $\hat{s}(\lambda, |C|)$ by $L(\lambda, |C|)$. By the efficiency property of the garbled circuit, we have $L(\lambda, |C|) = \text{poly}(\lambda, \hat{s}(\lambda, |C|)) = \text{poly}(\lambda, |C|)$.
3. An IBE scheme $\text{IBE} = (\text{IBE.Setup}, \text{IBE.Enc}, \text{IBE.KeyGen}, \text{IBE.Dec})$ with IND-CPA security whose identity space and message space are $\{0, 1\}^*$. We can instantiate IBE from various standard assumptions including LWE [3, 14], CDH , and Factoring [15].

Construction. Let $N := N(\lambda, R)$, $n := n(\lambda, R)$, and $t := t(\lambda, R)$ be the parameters associated with RDMPC. In the following construction, we run the protocol with $R = \lambda$. The basic CPFE scheme $\text{BCPFE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for the circuit class \mathcal{C}_ℓ works as follows. Note that the scheme below deviates from the syntax of bounded collusion FE because Setup and KeyGen take the collusion bound Q as binary form,

rather than unary form as defined in Remark 1. This change in syntax is to reflect the fact that these algorithms run in polylogarithmic time in Q rather than in polynomial time. Even with this change, we can consider the same correctness requirement and security notions for it.

Setup($1^\lambda, 1^\ell, Q$) : On input the security parameter λ , an input length $\ell = \text{poly}(\lambda)$ of the circuit family to be supported, and the upper bound for the collusion $1 \leq Q \leq 2^\lambda$ in *binary* form and compute $(\text{IBE.mpk}, \text{IBE.msk}) \leftarrow \text{IBE.Setup}(1^\lambda)$. Output the master key pair as $(\text{mpk}, \text{msk}) := (\text{IBE.mpk}, \text{IBE.msk})$.

KeyGen(msk, x, Q) : On input master secret key $\text{msk} = \text{IBE.msk}$, an input $x \in \{0, 1\}^\ell$ and the upper bound for the collusion $1 \leq Q \leq 2^\lambda$ in *binary* form and do the following:

1. Compute $\hat{x} \leftarrow \text{InpEnc}(1^\lambda, 1^\lambda, 1^\ell, x)$.
2. Sample $u \leftarrow [Q]$.
3. Sample random set $\Delta \subset [N]$ such that $|\Delta| = n$.
4. For all $j \in \Delta$ and $k \in [\hat{\ell}]$, generate a secret key as

$$\text{IBE.sk}_{u,j,k,\hat{x}_k} \leftarrow \text{IBE.KeyGen}(\text{IBE.msk}, (u, j, k, \hat{x}_k)),$$

where \hat{x}_k is the k -th bit of \hat{x} .

5. Output

$$\text{sk} = \left(u, \Delta, \{\text{IBE.sk}_{u,j,k,\hat{x}_k}\}_{j \in \Delta, k \in [\hat{\ell}]} \right). \quad (3.1)$$

Enc($\text{mpk}, C, 1^Q$) : On input the master public key $\text{mpk} = \text{IBE.mpk}$, a circuit $C \in \mathcal{C}_\ell$, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Compute $(\hat{C}_{i,1}, \dots, \hat{C}_{i,N}) \leftarrow \text{CktEnc}(1^\lambda, 1^\lambda, 1^\ell, C)$ for $i \in [Q]$.
2. Define the circuit $L_{i,j}(\cdot) := \text{Local}(\hat{C}_{i,j}, \cdot)$ with input length $\hat{\ell} := |\hat{x}|$. For all $i \in [Q]$ and $j \in [N]$, do the following:
 - (a) Run the garbling algorithm

$$\{\text{lab}_{i,j,k,b}\}_{k \in [\hat{\ell}], b \in \{0,1\}} \leftarrow \text{GC.Garble}(1^\lambda, L_{i,j}).$$

- (b) For all $k \in [\hat{\ell}]$ and $b \in \{0, 1\}$, compute

$$\text{IBE.ct}_{i,j,k,b} \leftarrow \text{IBE.Enc}(\text{IBE.mpk}, (i, j, k, b), \text{lab}_{i,j,k,b}).$$

3. Output

$$\text{ct} = \{\text{IBE.ct}_{i,j,k,b}\}_{i \in [Q], j \in [N], k \in [\hat{\ell}], b \in \{0,1\}}. \quad (3.2)$$

Dec($\text{ct}, \text{sk}, 1^Q$) : On input a secret key sk , a ciphertext ct , and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Parse the secret key as Eq. (3.1) and the ciphertext as Eq. (3.2).
2. For all $j \in \Delta$, do the following:

- (a) For all $k \in [\widehat{\ell}]$, compute $\text{lab}'_{u,j,k} := \text{IBE.Dec}(\text{IBE.sk}_{u,j,k,\widehat{x}_k}, \text{IBE.ct}_{u,j,k,\widehat{x}_k})$, where \widehat{x}_k is the k -th bit of \widehat{x} .
 - (b) Compute $\widehat{y}'_{u,j} := \text{GC.Eval}(\{\text{lab}'_{u,j,k}\}_{k \in [\widehat{\ell}]})$
3. Compute and output $z = \text{Decode}(\{\widehat{y}'_{u,j}\}_{j \in \Delta})$.

Remark 6 (Intuition for the construction.) The above scheme can be seen as a variant of the FE scheme by Ananth and Vaikuntanathan [12], who showed a generic construction of Q -bounded FE scheme from a single-key FE scheme. Here, we instantiate the single-key FE scheme with the construction by Sahai and Seyalioglu [28] and then replace the PKE used for encrypting the labels of the garbled circuits inside their construction with IBE. In more details, in our construction above, we run QN instances of the single-key FE scheme. These QN instances are grouped into Q groups each consisting of N instances. In the key generation algorithm of BCPFE, one chooses a group $u \in [Q]$ and then generates n out of N secret keys of the single-key FE instances in the group. To encrypt a message, one generates shares of the message (in our case, a circuit) using RDMPC and then encrypts them using N instances of FE for each group $i \in [Q]$. By the correctness of RDMPC, n secret keys of the single-key FE from a single group can recover the decryption results.

Security. The following theorem asserts the security of our CPFE scheme.

Theorem 3. *Assume that IBE satisfies IND-CPA security, GC is a secure garbled circuit scheme, and RDMPC is secure. Then, BCPFE for the circuit class \mathcal{C}_ℓ is NA-SIM-secure.*

We refer to the full version [6] for the detailed proof of Theorem 3.

3.3 A Variant of Basic Construction with AD-SIM Security

Here, we provide a variant of the basic construction in Sect. 3.2 that satisfies stronger AD-SIM security rather than NA-SIM security at the cost of only supporting circuits with bounded size. Similarly to the construction in Sect. 3.2, the construction can be upgraded into a construction with delayed collusion bound in Sect. 3.4. In more details, our construction is for the circuit class $\mathcal{C}_{\ell,s}$, where $\mathcal{C}_{\ell,s}$ is the set of circuits with input length ℓ and size at most s . Formally, our FE is for the relation $R_{\text{prm}} : \mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}} \rightarrow \{0, 1\}^*$ where $\text{prm} = (1^\ell, 1^s)$, $\mathcal{X}_{\text{prm}} := \mathcal{C}_{\ell,s}$, and $\mathcal{Y}_{\text{prm}} := \{0, 1\}^\ell$ and $R_{\text{prm}}(C, x) = C(x)$, for all $C \in \mathcal{C}_{\ell,s}$ and $x \in \{0, 1\}^\ell$.

Ingredients and Parameters. Our construction is the same as that in Sect. 3, but we require stronger SIM-RSO security for the IBE [25]. As shown in [25], IBE with SIM-RSO security can be constructed only from IBE with more standard IND-CPA security. Therefore, our construction here can be based on the same set of assumptions as the construction in Sect. 3. However, since the IBE scheme with SIM-RSO security can only encrypt messages with bounded length, we will have a bound on the length of the labels of GC. This in turn implies that we can only support circuits with bounded size as messages of the CPFE. In the construction, we encrypt circuits with fixed size $s = s(\lambda)$. We denote the size of the circuit $\text{Local}(\widehat{C}_j, \cdot)$ by \widehat{s} , where \widehat{C}_j is an output of

CktEnc on input a circuit C of size s . We also denote the length of the labels output by GC on input a circuit of size \hat{s} as L . While \hat{s} and L are polynomial function in λ and s , we treat them as functions only depend on λ here, since the size of the circuit s is fixed. We assume that the message space of IBE is $\{0, 1\}^L$.

We observe that Setup and KeyGen run in time $\text{poly}(\lambda, \ell, s, \log Q)$ by the efficiency properties of IBE and RDMPC. This means that these algorithms can be run even for super polynomial Q . Looking ahead, this property will be used crucially for the full-fledged construction that we show in Sect. 3.4. We also observe that Enc and Dec run in time $Q \cdot \text{poly}(\lambda, \ell, s)$ by the efficiency properties of IBE, GC, and RDMPC.

The following theorem asserts the security of our CPFE scheme.

Theorem 4. *Assume that IBE satisfies IND-CPA security and SIM-RSO security, GC is a secure garbled circuit scheme, and RDMPC is secure. Then, BCPFE for the circuit class $\mathcal{C}_{\ell, s}$ is AD-SIM-secure.*

Overview for the proof. Before going to the formal proof, we provide its overview. The proof is similar to that for Theorem 3. However, here, we have to consider secret key queries after the encryption query. Since we consider simulation-based security definition for BCPFE, this intuitively means that we have to be able to “program” a decryption result into a secret key issued after the encryption query. RDMPC already has this type of capability, since this is designed for constructing FE with AD-SIM security [12]. Using this property and the security of the garbled circuits, we can “program” the decryption results into labels of garbled circuits. What remains is to simulate the IBE ciphertexts and secret keys so that the decryption results correspond with the labels generated as above. For this purpose, we use IBE with SIM-RSO security [25].

We refer to the full version [6] for the detailed proof of Theorem 4.

3.4 Full-Fledged Construction

In Sects. 3.2 and 3.3, we construct bounded collusion CPFE schemes. Here, we show that these schemes can be converted into CPFE schemes with dynamic bounded collusion property. The resulting schemes satisfy the same level of the security and support the same class of circuits as the original schemes. The conversion is the same for both schemes. Let BCPFE = (BCPFE.Setup, BCPFE.KeyGen, BCPFE.Enc, BCPFE.Dec) be our FE scheme in either Sect. 3.2 or 3.3 and let \mathcal{C}_{prm} be the supported circuit class. We have $\text{prm} = 1^\ell$ or $\text{prm} = (1^\ell, 1^s)$. An input to a circuit $C \in \mathcal{C}_{\text{prm}}$ has fixed length $\ell(\lambda)$ in both cases.

Construction. We now construct a CPFE scheme CPFE = (CPFE.Setup, CPFE.KeyGen, CPFE.Enc, CPFE.Dec) with delayed collusion bound as follows.

Setup($1^\lambda, \text{prm}$) : On input the security parameter λ , the parameter prm that specifies the circuit family to be supported, do the following:

1. Run $(\text{BCPFE.mpk}_i, \text{BCPFE.msk}_i) \leftarrow \text{BCPFE.Setup}(1^\lambda, \text{prm}, 2^i)$ for $i \in [\lambda]$, where 2^i is represented as a binary number here.
2. Output $(\text{mpk}, \text{msk}) := (\{\text{BCPFE.mpk}_i\}_{i \in [\lambda]}, \{\text{BCPFE.msk}_i\}_{i \in [\lambda]})$.

KeyGen(msk, x) : On input master secret key $\text{msk} = \{\text{BCPFE.msk}_i\}_{i \in [\lambda]}$ and an input $x \in \{0, 1\}^\ell$, do the following:

1. Run $\text{BCPFE.sk}_i \leftarrow \text{BCPFE.KeyGen}(\text{IBE.msk}_i, x, 2^i)$ for $i \in [\lambda]$, where 2^i is represented as a binary number above.
 2. Output $\text{sk} := \{\text{BCPFE.sk}_i\}_{i \in [\lambda]}$.
- $\text{Enc}(\text{mpk}, C, 1^Q)$: On input the master public key mpk , a circuit $C \in \mathcal{C}_{\text{prm}}$, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:
1. Find u such that $2^{u-1} < Q \leq 2^u$.
 2. Parse $\text{mpk} \rightarrow \{\text{BCPFE.mpk}_i\}_{i \in [\lambda]}$.
 3. Compute $\text{BCPFE.ct}_u \leftarrow \text{BCPFE.Enc}(\text{BCPFE.mpk}_u, x, 1^{2^u})$ and output $\text{ct} = \text{BCPFE.ct}_u$.
- $\text{Dec}(\text{ct}, \text{sk}, 1^Q)$: On input a secret key sk , a ciphertext ct , and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:
1. Find u such that $2^{u-1} < Q \leq 2^u$.
 2. Parse $\text{sk} \rightarrow \{\text{BCPFE.sk}_i\}_{i \in [\lambda]}$ and $\text{ct} = \text{BCPFE.ct}_u$.
 3. Compute and output $\text{BCPFE.Dec}(\text{BCPFE.sk}_u, \text{BCPFE.ct}_u)$.

It is clear that the correctness of the above scheme follows from that of the underlying scheme BCPFE. We analyse the efficiency of the above construction in our full version [6].

The following theorem asserts the security of our CPFE scheme.

Theorem 5. *If BCPFE satisfies AD-SIM security as bounded FE scheme, then so does CPFE scheme as FE with delayed collusion bound. Similarly, if BCPFE satisfies NA-SIM security as bounded FE scheme, then so does CPFE scheme as FE with delayed collusion bound.*

The proof of the above theorem appears in our full version [6].

4 Succinct KPFE with Dynamic Bounded Collusion

In this section, we construct new succinct public key KPFE scheme. The construction substantially improves the security of the previous succinct bounded KPFE schemes [1, 20], because it supports delayed collusion bound and satisfies AD-SIM security. Furthermore, our construction can be instantiated only from the LWE assumption, similarly to the previous constructions.

In our full version [6], we also construct a CPFE scheme for unbounded size circuits with AD-SIM security and delayed collusion property. The construction is the dual version of the KPFE scheme in this section in the sense that it satisfies similar properties and the idea for the construction is very similar. Further, the scheme can be instantiated only from the LWE assumption, similar to the construction in this section.

We now describe the parameters for our KPFE scheme in more detail. We construct KPFE scheme for the circuit family $\mathcal{C}_{\text{inp,dep,out}}$ containing circuits with input length $\text{inp} = \text{inp}(\lambda)$, depth $\text{dep} = \text{dep}(\lambda)$, output length $\text{out} = \text{out}(\lambda)$ but with arbitrary polynomial size. Formally, we construct an FE scheme for $\text{prm} = (1^{\text{inp}}, 1^{\text{dep}}, 1^{\text{out}})$, $\mathcal{X}_{\text{prm}} = \{0, 1\}^{\text{inp}}$, $\mathcal{Y}_{\text{prm}} = \mathcal{C}_{\text{inp,dep,out}}$ and $R_{\text{prm}} : \mathcal{X}_{\text{prm}} \times \mathcal{Y}_{\text{prm}} \rightarrow \{0, 1\}^{\text{out}}$, where $R(x, C) = C(x)$, for all $C \in \mathcal{C}_{\text{inp,dep,out}}$ and $x \in \{0, 1\}^{\text{inp}}$.

Ingredients. The underlying building blocks for our KPFE construction are as follows:

1. A bounded, public key KPFE scheme, denoted by

$$1\text{KPFE} = (1\text{KPFE.Setup}, 1\text{KPFE.KeyGen}, 1\text{KPFE.Enc}, 1\text{KPFE.Dec}),$$

for the same functionality (i.e., R_{prm} defined above). We require the scheme to satisfy NA-SIM security against a *single* key query. We can instantiate such a scheme with the succinct KPFE scheme from LWE [20] (See Theorem 1).

2. A CPFE scheme denoted by

$$\text{CPFE} = (\text{CPFE.Setup}, \text{CPFE.KeyGen}, \text{CPFE.Enc}, \text{CPFE.Dec}),$$

for bounded polynomial sized circuits that already supports delayed collusion bound property and satisfies AD-SIM security. Namely, CPFE supports a circuit class $\mathcal{C}_{\text{inp}', \text{size}'}$ consisting of circuits with input length inp' and size at most size' , where the setting of the parameters is deferred until the description of the construction. Formally, we use FE with $\text{prm}' = 1^{\text{inp}'}$, $\mathcal{X}_{\text{prm}'} = \mathcal{C}_{\text{inp}', \text{size}'}$, $\mathcal{Y}_{\text{prm}'} = \{0, 1\}^{\text{inp}'}$ and $R_{\text{prm}'} : \mathcal{X}_{\text{prm}'} \times \mathcal{Y}_{\text{prm}'} \rightarrow \{0, 1\}^*$, where $R(C, x) = C(x)$, for all $C \in \mathcal{C}_{\text{inp}', \text{size}'}$ and $x \in \{0, 1\}^{\text{inp}'}$. We instantiate it from our construction in Sect. 3.4, which in turn can be based on any IBE.

3. A pseudorandom function PRF = (PRF.Setup, PRF.Eval). We assume without loss of generality that the input and output space of PRF is the 1KPFE public key space and the randomness space used in 1KPFE.Enc algorithm respectively.

At a high level, our KPFE construction is a compiler that applies the CPFE scheme already satisfying the delayed collusion property and AD-SIM security on top of the single key, succinct KPFE scheme. This makes the resultant KPFE satisfy the delayed collusion property and AD-SIM security as well. We now proceed to the formal construction below.

4.1 Construction

The KPFE scheme $\text{KPFE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ with delayed collusion bound for the circuit class $\mathcal{C}_{\text{inp}, \text{dep}, \text{out}}$ is as follows.

Setup($1^\lambda, \text{prm}$): On input the security parameter λ and the parameters $\text{prm} = (1^{\text{inp}}, 1^{\text{dep}}, 1^{\text{out}})$ do the following:

1. Compute $\text{prm}' := (1^{\text{inp}'}, 1^{\text{size}'})$, where $\text{inp}' := \text{inp}'(\lambda, \text{prm})$ is the length of the master public key for 1KPFE output by $1\text{KPFE.Setup}(1^\lambda, \text{prm})$ and $\text{size}' := \text{size}'(\lambda, \text{prm})$ is the size of the circuit $E_{[x, K]}$ described in Fig. 1.
2. Run $(\text{CPFE.mpk}, \text{CPFE.msk}) \leftarrow \text{CPFE.Setup}(1^\lambda, \text{prm}')$.
3. Output $(\text{mpk}, \text{msk}) := (\text{CPFE.mpk}, \text{CPFE.msk})$.

KeyGen(msk, C) : On input the master secret key $\text{msk} = \text{CPFE.msk}$ and a circuit $C \in \mathcal{C}_{\text{inp}, \text{dep}, \text{out}}$, do the following:

1. Compute $(1\text{KPFE.mpk}, 1\text{KPFE.msk}) \leftarrow 1\text{KPFE.Setup}(1^\lambda, \text{prm})$.
2. Generate a secret key under 1KPFE as $1\text{KPFE.sk} \leftarrow 1\text{KPFE.KeyGen}(1\text{KPFE.msk}, C)$.
3. Generate another secret key $\text{CPFE.sk} \leftarrow \text{CPFE.KeyGen}(\text{CPFE.msk}, 1\text{KPFE.mpk})$, where 1KPFE.mpk is interpreted as a string in $\{0, 1\}^{\text{inp}'}$.

4. Output the full secret key $sk := (\text{CPFE.sk}, 1\text{KPFE.sk})$.

$\text{Enc}(\text{mpk}, x, 1^Q)$: On input the master public key $\text{mpk} = \text{CPFE.mpk}$, an input $x \in \{0, 1\}^{\text{inp}}$ and the query bound $1 \leq Q < 2^\lambda$, do the following:

1. Sample a PRF key $K \leftarrow \text{PRF.Setup}(1^\lambda)$.
2. Using prm , construct the circuit $E_{[x, K]}(\cdot)$ defined as Fig. 1.
3. Compute a ciphertext $\text{CPFE.ct} \leftarrow \text{CPFE.Enc}(\text{CPFE.mpk}, E_{[x, K]}, 1^Q)$.
4. Output the ciphertext $\text{ct} := \text{CPFE.ct}$.

$\text{Dec}(\text{ct}, sk)$: On input a secret key sk associated with circuit C and a ciphertext ct , do the following:

1. Parse the ciphertext $\text{ct} = \text{CPFE.ct}$ and the secret key $sk = (\text{CPFE.sk}, 1\text{KPFE.sk})$, where CPFE.sk and 1KPFE.sk are associated with $1\text{KPFE.mpk} \in \{0, 1\}^{\text{inp}'}$ and the circuit C respectively.
2. Compute $y = \text{CPFE.Dec}(\text{CPFE.sk}, \text{CPFE.ct})$.
3. Compute and output $z = 1\text{KPFE.Dec}(1\text{KPFE.sk}, y)$.

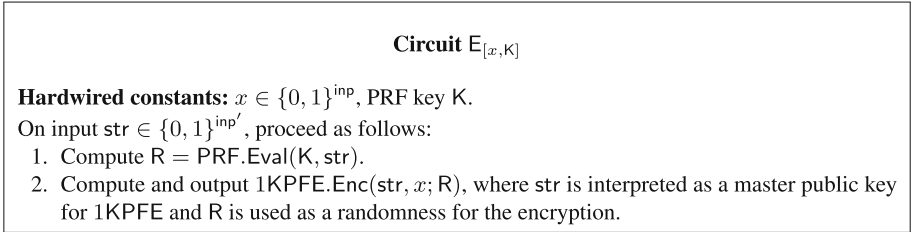


Fig. 1. Circuit $E_{[x, K]}$.

In our full version [6], we show that the scheme is correct, succinct and satisfies AD-SIM security.

5 FE for Turing Machines with Dynamic Bounded Collusion

In this section, we construct FE for Turing machines from KPFE and CPFE schemes that we have constructed so far. For conciseness, we first provide a generic construction of FE that combines many instance of FE together in Sect. 5.1. We then instantiate this generic construction to construct FE for Turing machines with NA-SIM security in Sect. 5.2 and FE for NL with AD-SIM security in our full version [6].

5.1 Generalized Bundling of Functionality

Consider an FE scheme $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ for a parameter $\text{prm} = 1^i$ and a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \rightarrow \{0, 1\} \cup \{\perp\}$ for all $i \in \mathbb{N}$. Using such FE, we will construct a new FE with message space \mathcal{A} and key space \mathcal{B} . We assume that there exist efficiently computable maps $\mathcal{S} : \mathbb{N} \rightarrow 2^{\mathbb{N}}$ and $\mathcal{T} : \mathbb{N} \rightarrow 2^{\mathbb{N}}$ such

that $\max \mathcal{S}(n)$ and $\max \mathcal{T}(n)$ can be bounded by some fixed polynomial in n . We also assume that there exist maps f with domain \mathcal{A} and g with domain \mathcal{B} such that

$$f(x) \in \prod_{i \in \mathcal{S}(|x|)} \mathcal{X}_i \quad \text{and} \quad g(y) \in \prod_{i \in \mathcal{T}(|y|)} \mathcal{Y}_i,$$

where $|x|$ and $|y|$ are the lengths of x and y as binary strings. Namely, f and g are maps such that

$$f : \mathcal{A} \ni x \mapsto \{f(x)_i \in \mathcal{X}_i\}_{i \in \mathcal{S}(|x|)}, \quad g : \mathcal{B} \ni y \mapsto \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}.$$

Here, we require that the length of $|f(x)|_i$ and $|g(y)|_i$ can be computed from the length of $|x|$ alone and they do not depend on the actual value of x . In this setting, we can construct an FE scheme $\text{BFE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for a two input function $R^{\text{bndl}} : \mathcal{A} \times \mathcal{B} \rightarrow \{0, 1\}^*$ defined in the following

$$R^{\text{bndl}}(x, y) = \{R_i(f(x)_i, g(y)_i)\}_{i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)}, \tag{5.1}$$

where $f(x)_i \in \mathcal{X}_i$ and $g(y)_i \in \mathcal{Y}_i$ are the i -th entries of $f(x)$ and $g(y)$, respectively.

Ingredients. We now describe the underlying building blocks used to obtain our FE construction:

1. A pseudorandom function $\text{PRF} = (\text{PRF.Setup}, \text{PRF.Eval})$. We assume that $\text{PRF.Eval}(K, \cdot)$ has domain $\{0, 1\}^\lambda$ and range $\{0, 1\}^\lambda$ for any key K output by $\text{PRF.Setup}(1^\lambda)$. The input space $\{0, 1\}^\lambda$ can be regarded as $[2^\lambda]$ by the natural bijection between the two sets. We can instantiate PRF from any one-way function [18].
2. An FE scheme $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ for a parameter $\text{prm} = 1^i$ and a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \rightarrow \{0, 1\} \cup \{\perp\}$ for $i \in \mathbb{N}$ with delayed collusion property. We require the scheme to have either NA-SIM or AD-SIM security. We also require that the randomness used by $\text{FE.Setup}(1^\lambda, 1^i)$ is of fixed length λ . This is without loss of generality, since we can generate unbounded length of pseudorandom bits from a binary string R of length λ by regarding R as a PRF key.
3. A garbled circuit scheme $\text{GC} = (\text{GC.Garble}, \text{GC.Eval})$. We assume that a label is represented by a binary string and denote its length by $L(\lambda, |C|)$, where C is the circuit being garbled. We can instantiate it by Yao’s garbled circuit [30], which can be based on any one-way function.
4. An IBE scheme $\text{IBE} = (\text{IBE.Setup}, \text{IBE.Enc}, \text{IBE.KeyGen}, \text{IBE.Dec})$ with IND-CPA security whose identity space and message space are $\{0, 1\}^*$. We assume that the key generation algorithm is deterministic. This is without loss of generality, since we can use PRF to derandomize the key generation algorithm. We can instantiate IBE from various standard assumptions including LWE [3, 14], CDH , and Factoring [15].

Construction. We then provide the description of the construction of $\text{BFE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for R^{bndl} above.

Setup(1^λ) : On input the security parameter λ , do the following:

1. Run $(\text{IBE.mpk}, \text{IBE.msk}) \leftarrow \text{IBE.Setup}(1^\lambda)$.
2. Sample a PRF key $K \leftarrow \text{PRF.Setup}(1^\lambda)$.
3. Output the master key pair as $(\text{mpk}, \text{msk}) := (\text{IBE.mpk}, (\text{IBE.msk}, K))$.

KeyGen(msk, y) : On input master secret key $\text{msk} = \text{IBE.msk}$, a key attribute $y \in \mathcal{B}$, do the following:

1. Compute $\mathcal{T}(|y|) \subseteq \mathbb{N}$, where $|y|$ is the length of y as a binary string.
2. Compute $R_i = \text{PRF.Eval}(K, i)$ for $i \in \mathcal{T}(|y|)$, where $i \in \mathbb{N}$ is interpreted as a binary string in $\{0, 1\}^\lambda$.
3. Run $(\text{FE.mpk}_i, \text{FE.msk}_i) \leftarrow \text{FE.Setup}(1^\lambda, 1^i; R_i)$ for $i \in \mathcal{T}(|y|)$.
4. Compute $g(y) = \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}$.
5. For $i \in \mathcal{T}(|y|)$, compute

$$\text{FE.sk}_i \leftarrow \text{FE.KeyGen}(\text{FE.msk}_i, g(y)_i).$$

6. Let $\ell_i := |\text{FE.mpk}_i|$. For all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$, generate a secret key as

$$\text{IBE.sk}_{i,j} \leftarrow \text{IBE.KeyGen}(\text{IBE.msk}, (i, j, \text{FE.mpk}_{i,j}))$$

where $\text{FE.mpk}_{i,j}$ is the j -th bit of $\text{FE.mpk}_i \in \{0, 1\}$ as a binary string.

7. Output

$$\text{sk} = \left(\mathcal{T}(|y|), \left\{ \text{FE.sk}_i, \left\{ \text{IBE.sk}_{i,j} \right\}_{j \in [\ell_i]} \right\}_{i \in \mathcal{T}(|y|)} \right). \quad (5.2)$$

Enc($\text{mpk}, x, 1^Q$) : On input the encryption key $\text{mpk} = \text{IBE.mpk}$, a message $x \in \mathcal{A}$, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Compute $\mathcal{S}(|x|) \subset \mathbb{N}$, where $|x|$ is the length of x as a binary string.
2. Compute $f(x) = \{f(x)_i\}_{i \in \mathcal{S}(|x|)}$.
3. Do the following for $i \in \mathcal{S}(|x|)$.
 - (a) Compute the length ℓ_i of FE.mpk_i . This is done without knowing FE.mpk_i .
 - (b) Sample a randomness r_i for the encryption algorithm $\text{FE.Enc}(\text{FE.mpk}_i, f(x)_i, 1^Q; r_i)$. This is done without knowing FE.mpk_i .
 - (c) Define a circuit

$$E_i(\cdot) := \text{FE.Enc}(\cdot, f(x)_i, 1^Q; r_i)$$

that takes as input a string $\text{str} \in \{0, 1\}^{\ell_i}$ and outputs $\text{FE.Enc}(\text{str}, f(x)_i, 1^Q; r_i)$, where str is interpreted as a master public key of the FE.

- (d) Generate a garbled circuit

$$\{\text{lab}_{i,j,b}\}_{j \in [\ell_i], b \in \{0,1\}} \leftarrow \text{GC.Garble}(1^\lambda, E_i).$$

- (e) For all $j \in [\ell_i]$ and $b \in \{0, 1\}$, compute

$$\text{IBE.ct}_{i,j,b} \leftarrow \text{IBE.Enc}(\text{IBE.mpk}, (i, j, b), \text{lab}_{i,j,b}).$$

4. Output

$$\text{ct} = \left(\mathcal{S}(|x|), \left\{ \text{IBE.ct}_{i,j,b} \right\}_{i \in \mathcal{S}(|x|), j \in [\ell_i], b \in \{0,1\}} \right). \quad (5.3)$$

$\text{Dec}(\text{ct}, \text{sk})$: On input a secret key sk , a ciphertext ct , and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Parse the secret key as Eq. (5.2) and the ciphertext as Eq. (5.3).
2. For all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ do the following.
 - (a) Retrieve FE.mpk_i from the FE.sk_i .
 - (b) For all $j \in [\ell_i]$ compute $\text{lab}'_{i,j} := \text{IBE.Dec}(\text{IBE.sk}_{i,j}, \text{FE.mpk}_{i,j}, \text{IBE.ct}_{i,j}, \text{FE.mpk}_{i,j})$.
 - (c) Compute $c_i := \text{GC.Eval}(\{\text{lab}'_{i,j}\}_{j \in [\ell_i]})$.
 - (d) Compute $z_i := \text{FE.Dec}(\text{FE.sk}_i, c_i)$
3. Output $\{z_i\}_{i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)}$.

In the full version [6], we show that the scheme is correct, satisfies efficiency in that all the algorithms run in time polynomial in their respective input lengths and satisfies AD-SIM (resp., NA-SIM) security, if the same holds for the FE.

5.2 FE for Turing Machines with NA-SIM Security

Here, we provide the construction of FE for Turing machines defined as in Sect. 2.1.3, which satisfies NA-SIM security. Recall that in FE for Turing machines, a ciphertext is associated with $(x, 1^t)$ and a secret key is for a Turing machine M , and the decryption results to 1 if the machine accepts the input within t steps and 0 otherwise. To construct such a scheme, we start with constructing two schemes with partial functionality and then combine them. The one scheme takes care of the case where $|(x, 1^t)| > |M|$, while the other takes care of the case where $|(x, 1^t)| \leq |M|$. Both schemes are obtained by applying the generic conversion in Sect. 5.1 to the schemes we constructed so far.

5.2.1 The Case of $|(x, 1^t)| > |M|$

We first show that by applying the conversion in Sect. 5.1 to the CPFSE scheme in Sect. 3.2, we can obtain an FE scheme for Turing machines for the case where $|(x, 1^t)| > |M|$. Formally, we construct an FE for $R^> : \mathcal{A} \times \mathcal{B} \rightarrow \{0, 1\}$, where $\mathcal{A} = \{0, 1\}^*$, \mathcal{B} is the set of all Turing machines, and

$$R^>((x, 1^t), M) = \begin{cases} 1 & \text{(if } M \text{ accepts } x \text{ in } t \text{ steps)} \wedge (|(x, 1^t)| > |M|) \\ 0 & \text{otherwise.} \end{cases}$$

To apply the conversion, we recall that the scheme in Sect. 3.2 is an FE for $\text{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \rightarrow \{0, 1\}$ where \mathcal{X}_i is the set of circuits with input length i , $\mathcal{Y}_i = \{0, 1\}^i$, and $R_i(C, x) = C(x)$. We then set \mathcal{S} , \mathcal{T} , f , and g as

$$\mathcal{S}(i) = \{1, 2, \dots, i - 1\}, \quad \mathcal{T}(i) = \{i\}, \quad f(x, 1^t) = \{U_{i,x,t}(\cdot)\}_{i \in [|x, 1^t|] - 1}, \quad g(M) = M$$

where $U_{i,x,t}(\cdot)$ is defined as Fig. 2. The circuit (in particular, Step 2 of the computation) is padded so that the circuit size only depends on $|(x, 1^t)|$. Note that $U_{i,x,t}$ is in \mathcal{X}_i even for x and t with unbounded length, since \mathcal{X}_i contains circuits of unbounded size.

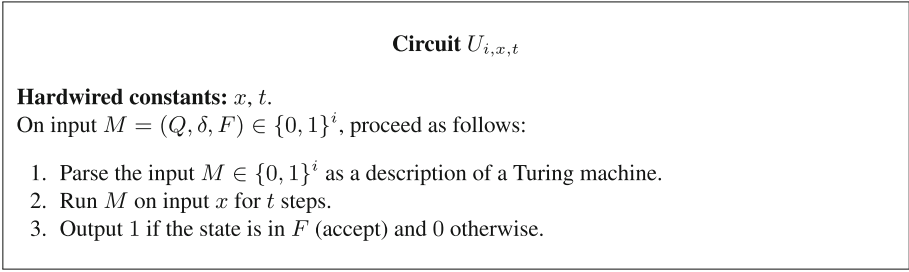


Fig. 2. Circuit $U_{i,x,t}$.

Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathcal{A}, \mathcal{B}$ defined as above, R^{bndl} defined as Eq. (5.1) is equivalent to $R^>$ except for the case of $|(x, 1^t)| \leq |M|$. In this case, the decryption result in FE for R^{bndl} is an empty set \emptyset , whereas it should be 0 in FE for $R^>$. However, the former can be converted into the latter very easily. To do so, we add the extra step to the decryption algorithm of the former where it outputs 0 if the decryption result is \emptyset . Since the original scheme in Sect. 3.2 is NA-SIM secure, so is the resulting scheme by the security of our bundling construction in Sect. 5.1.

5.2.2 The Case of $|(x, 1^t)| \leq |M|$

We next show that by applying the conversion in Sect. 5.1 to the KPFE scheme in Sect. 4, we can obtain an FE scheme for Turing machines for the case where $|(x, 1^t)| \leq |M|$. Formally, we construct an FE for $R^{\leq} : \mathcal{A} \times \mathcal{B} \rightarrow \{0, 1\}$, where $\mathcal{A} = \{0, 1\}^*$, \mathcal{B} is the set of all Turing machines, and

$$R^{\leq}((x, 1^t), M) = \begin{cases} 1 & \text{(if } M \text{ accepts } x \text{ in } t \text{ steps)} \wedge (|(x, 1^t)| \leq |M|) \\ 0 & \text{otherwise.} \end{cases}$$

To apply the conversion, we observe that the scheme in Sect. 4 can be used as an FE for $\text{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \rightarrow \{0, 1\}$ where \mathcal{X}_i is the set of circuits with input length i , depth $i \cdot \lambda$, and output length 1 and $\mathcal{Y}_i = \{0, 1\}^i$, and $R_i(C, x) = C(x)$. We then set $\mathcal{S}, \mathcal{T}, f$, and g as

$$\mathcal{S}(i) = i, \quad \mathcal{T}(i) = \{1, 2, \dots, i\}, \quad f(x, 1^t) = (x, 1^t), \quad g(M) = \{U_{i,M}(\cdot)\}_{i \in [M]},$$

where $U_{i,M}(\cdot)$ is defined as Fig. 3. Here, we check that $U_{i,M}(\cdot)$ is in \mathcal{Y}_i . Recall that even though \mathcal{Y}_i supports circuits with unbounded size, it has a bound on the depth of the circuit. We therefore argue that the depth of $U_{i,M}(\cdot)$ does not exceed $i\lambda$, even for unbounded size $|M|$. We evaluate the depth of Step 2 of the circuit, since this is the only non-trivial step. In the full version [6], we prove that this step can be implemented by a circuit with depth

$$t \cdot \text{poly}(\log |x|, \log t, \log |M|) \leq i \cdot \text{poly}(\log \lambda) \leq i \cdot \lambda$$

Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathcal{A}, \mathcal{B}$ defined as above, R^{bndl} defined as Equation (5.1) is equivalent to R^{\leq} except for the case of $|(x, 1^t)| > |M|$. In this case, the decryption result in FE for R^{bndl} is an empty set \emptyset , whereas it should be 0 in FE for $R^>$. However, the former can be converted into the latter by adding the extra step to the decryption algorithm where it outputs 0 if the decryption result is \emptyset . This gives us the construction of FE for R^{\leq} . Since the original scheme is AD-SIM secure, so is the resulting scheme by the security of our bundling construction in Sect. 5.1.

5.2.3 Putting the Pieces Together

Here, we combine the two schemes we considered so far to obtain the full-fledged scheme. We set $\mathcal{A} = \{0, 1\}^*$ and \mathcal{B} to be the set of all Turing machines. We also set $R_1 = R^>$, $R_2 = R^{\leq}$, $\mathcal{X}_i = \mathcal{A}$, $\mathcal{Y}_i = \mathcal{B}$ for $i = 1, 2$. We have already constructed schemes for R_1 and R_2 and now combine them. To do so, we set $\mathcal{S}, \mathcal{T}, f$, and g as

$$\mathcal{S}(i) = \{1, 2\}, \quad \mathcal{T}(i) = \{1, 2\}, \quad f(x, 1^t) = \{(x, 1^t), (x, 1^t)\}, \quad g(M) = \{M, M\}$$

We observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathcal{A}, \mathcal{B}$ defined as above, R^{bndl} defined as Equation (5.1) is

$$R^{\text{bndl}}((x, 1^t), M) = \begin{cases} (1, 0) & \text{(if } M \text{ accepts } x \text{ in } t \text{ steps) } \wedge (|(x, 1^t)| > |M|) \\ (0, 1) & \text{(if } M \text{ accepts } x \text{ in } t \text{ steps) } \wedge (|(x, 1^t)| \leq |M|) . \\ (0, 0) & \text{otherwise.} \end{cases}$$

An FE for the above relation is not exactly the same as the FE for Turing machines we defined in Sect. 2.1.3. However, the former readily implies the latter. To do so, we add the extra step to the decryption algorithm of the former where it checks whether there is 1 in the left or right slot of the decryption result and outputs 1 if there is. Otherwise, it outputs 0.

It is obvious that the obtained scheme satisfies correctness and efficiency requirements if so does the underlying schemes. As for the security, we can see by the security of our bundling construction in Sect. 5.1 that the resulting scheme is NA-SIM secure if we combine an AD-SIM and an NA-SIM secure scheme for R^{\leq} and $R^>$ respectively.

Circuit $U_{i,M}$

Hardwired constants: Description of a Turing Machine M .
 On input $y \in \{0, 1\}^i$, proceed as follows:

1. Parse the input $y \in \{0, 1\}^i$ as $(x, 1^t)$.
2. Otherwise, run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Fig. 3. Circuit $U_{i,M}$.

Remark 7. One might think that FE for R^{bndl} constructed above leaks more information on $(x, 1^t)$ than FE for Turing machines and so is our final scheme, because the decryption result can be \emptyset , in addition to 0 or 1 and the decryptor can know whether $|(x, 1^t)| > |M|$ or not. However, it is not the case since whether the decryption result is \emptyset or not can be checked only from the length of the string $|(x, 1^t)|$, which is not meant to be hidden in our definition (and other standard definitions) of FE.

To sum up, we have the following theorem:

Theorem 6. *We have FE for Turing machines with delayed collusion property that satisfies NA-SIM security from the sub-exponential LWE assumption.*

Due to space constraints, we provide our AD-SIM secure FE for NL in the full version [6].

Acknowledgement. The fourth author was supported by JSPS KAKENHI Grant Number 19H01109 and JST CREST JPMJCR19F6.

References

1. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 3–35. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_1
2. Agrawal, S.: Indistinguishability obfuscation without multilinear maps: new methods for bootstrapping and instantiation. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 191–225. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_7
3. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
4. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_28
5. Agrawal, S., Maitra, M.: FE and iO for turing machines from minimal assumptions. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 473–512. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_18
6. Agrawal, S., Maitra, M., Vempati, N.S., Yamada, S.: Functional encryption for turing machines with dynamic bounded collusion from LWE. Cryptology ePrint Archive Report 2021/848 (2021). <https://eprint.iacr.org/2021/848>
7. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from LWE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 765–797. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_26
8. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 173–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_7
9. Agrawal, S., Singh, I.P.: Reusable garbled deterministic finite automata from learning with errors. In: ICALP (2017)

10. Ananth, P., Jain, A., Lin, H., Matt, C., Sahai, A.: Indistinguishability obfuscation without multilinear maps: $i\mathcal{O}$ from LWE, bilinear maps, and weak pseudorandomness. In: *Crypto* (2019)
11. Ananth, P., Sahai, A.: Functional encryption for turing machines. In: Kushilevitz, E., Malkin, T. (eds.) *TCC 2016*. LNCS, vol. 9562, pp. 125–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_6
12. Ananth, P., Vaikuntanathan, V.: Optimal bounded-collusion secure functional encryption. In: Hofheinz, D., Rosen, A. (eds.) *TCC 2019*. LNCS, vol. 11891, pp. 174–198. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_8
13. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) *TCC 2011*. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
14. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) *EUROCRYPT 2010*. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
15. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) *CRYPTO 2017*. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18
16. Garg, R., Goyal, R., Lu, G., Waters, B.: Dynamic collusion bounded functional encryption from identity-based encryption. *Personal Communication* (2021)
17. Gay, R., Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from simple-to-state hard problems: new assumptions, new techniques, and simplification. In: *STOC* (2021)
18. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: *FOCS* (1984)
19. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) *CRYPTO 2013*. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_30
20. Goldwasser, S., Tauman Kalai, Y., Popa, R., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: *STOC* (2013)
21. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) *CRYPTO 2012*. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
22. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) *TCC 2016*. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14
23. Jain, A., Lin, H., Matt, C., Sahai, A.: How to leverage hardness of constant-degree expanding polynomials over \mathbb{R} to build $i\mathcal{O}$. In: Ishai, Y., Rijmen, V. (eds.) *EUROCRYPT 2019*. LNCS, vol. 11476, pp. 251–281. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_9
24. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. *Cryptology ePrint Archive Report 2020/1003* (2020)
25. Kitagawa, F., Tanaka, K.: Key dependent message security and receiver selective opening security for identity-based encryption. In: Abdalla, M., Dahab, R. (eds.) *PKC 2018*. LNCS, vol. 10769, pp. 32–61. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_2
26. Lin, H., Luo, J.: Compact adaptively secure ABE from k -lin: beyond NC1 and towards NL. In: *EUROCRYPT* (2020)
27. O’Neill, A.: Definitional issues in functional encryption. *Cryptology ePrint Archive Report 2010/556* (2010)

28. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: CCS (2010)
29. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
30. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS (1982)



Receiver-Anonymity in Rerandomizable RCCA-Secure Cryptosystems Resolved

Yi Wang¹, Rongmao Chen¹(✉), Guomin Yang², Xinyi Huang³(✉),
Baosheng Wang¹, and Moti Yung^{4,5}

¹ School of Computer, National University of Defense Technology, Changsha, China
{wangyi14, chromao, bswang}@nudt.edu.cn

² Institute of Cybersecurity and Cryptology, School of Computing and Information
Technology, University of Wollongong, Wollongong, NSW 2522, Australia
gyang@uow.edu.au

³ Fujian Provincial Key Laboratory of Network Security and Cryptology, College
of Mathematics and Informatics, Fujian Normal University, Fuzhou, China
xyhuang@fjnu.edu.cn

⁴ Google LLC, New York, NY, USA

⁵ Columbia University, New York, USA
moti@cs.columbia.edu

Abstract. In this work we resolve the open problem raised by Prabhakaran and Rosulek at CRYPTO 2007, and present the *first* anonymous, rerandomizable, Replayable-CCA (RCCA) secure public-key encryption scheme. This solution opens the door to numerous privacy-oriented applications with a highly desired RCCA security level. At the core of our construction is a non-trivial extension of smooth projective hash functions (Cramer and Shoup, EUROCRYPT 2002), and a modular generic framework developed for constructing rerandomizable RCCA-secure encryption schemes with receiver-anonymity. The framework gives an enhanced abstraction of the original Prabhakaran and Rosulek's scheme (which was the first construction of rerandomizable RCCA-secure encryption in the standard model), where the most crucial enhancement is the first realization of the desirable property of receiver-anonymity, essential to privacy settings. It also serves as a conceptually more intuitive and generic understanding of RCCA security, which leads, for example, to new implementations of the notion. Finally, note that (since CCA security is not applicable to the privacy applications motivating our work) the concrete results and the conceptual advancement presented here, seem to substantially expand the power and relevance of the notion of rerandomizable RCCA-secure encryption.

Keywords: RCCA security · Receiver-anonymity · Smooth projective hash function

1 Introduction

RCCA security. Security against adaptive chosen-ciphertext attacks (CCA) is widely considered as a *de facto* security standard for public-key encryption

(PKE). However, it is evidenced that for some practical purposes, a somewhat weaker security notion than CCA security is already sufficient [1, 16, 25]. To this end, Canetti et al. [5] introduced the notion of Replayable-CCA (RCCA) security, which is essentially the same as CCA security, except that no guarantees are given against adversaries with the capability of malleating a ciphertext into a new one of the same plaintext. Such a relaxation endows PKE with desirable features such as rerandomizable RCCA (Rand-RCCA) security which was proposed by Canetti et al. [5] and later formalized by Groth [14]. This notion turns out to have numerous practical applications, such as: cryptographic reverse firewalls [9, 12, 18], mixnets [13, 20] and controlled-malleable NIZK [11].

Constructing Rand-RCCA-secure PKE has been generally considered a difficult problem, and was posed as an open problem in [5]. The difficulty is mainly due to the fact that RCCA security and rerandomizability are seemingly incompatible in some sense. In particular, the construction has to be almost CCA secure while at the same time has special mathematical structure for realizing rerandomizability. A notable construction was by Prabhakaran and Rosulek [22] at CRYPTO 2007 (hereafter referred to as PR scheme) which is the first perfect Rand-RCCA-secure PKE based on the DDH assumption in the standard model.

Receiver-anonymity in the RCCA setting. In [22], Prabhakaran and Rosulek further defined a new notion called *RCCA receiver-anonymity* which is similar to the notion of *key-privacy* introduced by Bellare et al. in [2] but in the RCCA setting. For an RCCA receiver-anonymous encryption scheme, the generated ciphertext should not tell the adversary any information about the underlying public key. Such a property turns out to be essential in privacy-oriented applications where ciphertext-rerandomizability, adaptive security (i.e., permitting strong adversary who may probe the system with ciphertexts), and receiver-anonymity are required simultaneously.

A typical example—given by Prabhakaran and Rosulek [22]—is the application of rerandomizable encryption in mixnets where receiver-anonymity is indispensable. More precisely, consider an anonymous communication (AC) protocol based on universal mixnet [13] where a set of message relays (called mixnodes or mixes) receive a batch of encrypted messages, rerandomize and randomly permute them, and send them on their way forward. Unfortunately, the requirement of ciphertext-rerandomizability, while enabling unlinkability of multiple ciphertexts in terms of their contents, contradicts the desirable strong CCA security. Thus, as it turned out, only rerandomizable CPA-secure encryption schemes are used in previous universal mixnet-based AC protocols [13]. To strengthen the security to the adaptive one (i.e., allowing an adversary of the network to attempt sending ciphertexts of its own to the network as part of its attack), RCCA security is the alternative as it reconciles the required rerandomizability and adaptive security (this active attacker, in fact, is what most earlier works on anonymity are not protected against due to the encryption being CPA-secure only). However, as pointed out by Prabhakaran and Rosulek, without receiver-anonymity, the attacker might still be able to correlate the ciphertexts for the same recipient (i.e., sender-receiver relationships are not broken by the

mixing!). This example application demonstrates that anonymous Rand-RCCA-secure PKE is meaningful to strengthening the security of universal mixnet-based AC protocol on the one hand, and to allowing it to achieve anonymity (breaking completely sender-receiver relationship) at the same time. More broadly, for various other privacy-oriented applications [19, 23, 24, 28], RCCA receiver-anonymity is also desirable for privacy protection while withstanding strong adversary with decryption query capability (see the full version [29] for further motivating applications).

The open problem. Unfortunately, the PR scheme [22] does not achieve receiver-anonymity, and therefore, how to construct an anonymous Rand-RCCA-secure PKE to support the above mentioned applications under strong adversary was left as an explicit open problem by Prabhakaran and Rosulek in [22]:

“Adding anonymity brings out the power of rerandomizability and yields a potent cryptographic primitive. We note that our scheme does not achieve this definition of anonymity, and leave it as an interesting open problem.”

Somewhat surprisingly, in spite of further developments in constructing Rand-RCCA encryption throughout many years [6, 10, 11, 13, 14, 17, 22], the above open problem remains unsolved to date. The main technical challenge of achieving RCCA receiver-anonymity arises from the fact that different from the typical CCA game, the decryption oracle in the RCCA game would output “replay” if the query decryption result equals to either of the challenge plaintexts. Such a relaxation, in fact, gives the adversary more power and consequently raises the difficulty to achieve receiver-anonymity in the RCCA setting. Specifically, the adversary can guess the underlying public key, re-encrypt the challenge ciphertext and verify its guess via querying the decryption oracle. Thus, to defend against this attack, it is required that the rerandomization of ciphertext should not involve the public key. Such a feature was originally referred to as “universal rerandomization” by Golle et al. [13]. However, achieving receiver-anonymity is more challenging than realizing universal rerandomizability, since there may exist other ways allowing the adversary to rerandomize a ciphertext using the public key. In other words, receiver-anonymity is strictly stronger than universal rerandomizability. An example is the PR scheme which is universally rerandomizable but not receiver-anonymous (see Sect. 2 for the detailed analysis).

Motivated by the aforementioned state of affairs and the requirement of receiver-anonymity for privacy-oriented applications, our main goal in this work is to resolve the above challenging problem of achieving RCCA receiver-anonymity. More specifically, we ask *whether it is possible to achieve receiver-anonymity in the RCCA setting; and if the answer is positive, how to attempt a solution which is as generic as possible*. Our second question is motivated by the fact that a generic paradigm would enable a better understanding of the underlying key ideas and more diversified constructions of anonymous Rand-RCCA-secure encryption in a conceptually clear and modular way. Also, a framework using abstract building blocks enables more concrete instantiations from various assumptions, leading to better security (as will be demonstrated by our additional results below).

Our results. We resolve the Prabhakaran and Rosulek’s open problem in this work. We design a modular framework for constructing anonymous Rand-RCCA-secure PKE via an extension of the notion of smooth projective hash functions by Cramer and Shoup [8]. Our contributions can be summarized as follows:

- We formalize a novel extension of smooth projective hash function with various types of rerandomizability (Re-SPHF), and redefine the property of smoothness which is crucial to generally realize Rand-RCCA security with receiver-anonymity;
- We design a framework for constructing anonymous Rand-RCCA-secure PKE from Re-SPHFs, and rigorously prove its RCCA security and receiver-anonymity. These turn out to provide a conceptually intuitive understanding of RCCA security and receiver-anonymity;
- We provide the *first* anonymous Rand-RCCA-Secure PKE scheme from k -linear (k -LIN) assumption, which—putting anonymity aside—also improves the PR scheme with its more general hardness assumption.

Remark. It is worth noting that in [22], Prabhakaran and Rosulek also pointed out the potential of generalizing their scheme by following the Cramer-Shoup paradigm [8] (hereafter referred to as CS-paradigm), but they left such an investigation open as well. In fact, as we will illustrate in this work, our proposed framework can, in fact, be viewed as an abstraction of a modified PR scheme. Thus, while mainly motivated by achieving a solution to the RCCA receiver-anonymity, our work also closes Prabhakaran and Rosulek’s second open question of generalization via SPHFs.

2 Technical Overview and Related Work

First, let us explain why the PR scheme does not satisfy receiver-anonymity. As a countermeasure, we introduce a concrete approach to achieving RCCA receiver-anonymity based on the PR scheme. To generalize our proposed approach, following the SPHF-based CS-paradigm [8], we then define an extension of SPHF that could well explain the modified PR scheme and its security. To this end, we successfully design a general framework for anonymous, Rand-RCCA-secure PKE, which can, in turn, be instantiated based on different assumptions.

Why the PR scheme is not receiver-anonymous? We start by reviewing the PR scheme and its core idea leading to the RCCA security. The crucial idea toward achieving this goal is using two “strands” of Cramer-Shoup ciphertexts [8] which can be “uniquely” recombined with each other for rerandomization without changing the underlying plaintext.

Overview of the PR scheme. Let \mathbb{G} , $\overline{\mathbb{G}}$ be two cyclic groups of prime orders p , q where $p = 2q + 1$ where $\overline{\mathbb{G}}$ is also a subgroup of \mathbb{Z}_p^* . Let g and \bar{g} be generators of \mathbb{G} and $\overline{\mathbb{G}}$ respectively, \mathbf{a} denotes vector $(g^{a_1}, \dots, g^{a_n})$ for $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{Z}_p^n$,

and $\overline{\mathbf{a}}$ denotes vector $(\overline{g}^{a_1}, \dots, \overline{g}^{a_n})$ for $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{Z}_q^n$. The ciphertext of the PR scheme is

$$\zeta := \left(\underbrace{[u(\mathbf{x} + \mathbf{z})], M \cdot [\mathbf{b}^\top \mathbf{x}], [\boldsymbol{\alpha}^\top \mathbf{x}]}_{C_1: \text{ message-carrying strand}}, \underbrace{[u\mathbf{y}], [\mathbf{b}^\top \mathbf{y}], [\boldsymbol{\alpha}^\top \mathbf{y}]}_{C_2: \text{ rerandomization strand}}, \varrho \right) \tag{1}$$

$$\varrho := \left(\underbrace{[\overline{\mathbf{x}}], u \cdot [\overline{\mathbf{b}}^\top \overline{\mathbf{x}}], [\overline{\mathbf{c}}^\top \overline{\mathbf{x}}]}_{C_3: \text{ mask-carrying strand}}, \underbrace{[\overline{\mathbf{y}}], [\overline{\mathbf{b}}^\top \overline{\mathbf{y}}], [\overline{\mathbf{c}}^\top \overline{\mathbf{y}}]}_{C_4: \text{ rerandomization strand}} \right)$$

where $u \in \overline{\mathbb{G}}$, given fixed $\mathbf{g} \in \mathbb{Z}_p^4$ and $\overline{\mathbf{g}} \in \mathbb{Z}_q^2$, $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^4$ with $\mathbf{x} = x\mathbf{g}$, $\mathbf{y} = y\mathbf{g}$ for $x, y \in \mathbb{Z}_p$ and $\mathbf{z} \neq z\mathbf{g}$ for any $z \in \mathbb{Z}_p$, $\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{b}}, \overline{\mathbf{c}} \in \mathbb{Z}_q^2$ with $\overline{\mathbf{x}} = \overline{x}\overline{\mathbf{g}}$, $\overline{\mathbf{y}} = \overline{y}\overline{\mathbf{g}}$ for $\overline{x}, \overline{y} \in \mathbb{Z}_q$, $\boldsymbol{\alpha} = \mathbf{c} + \tau\mathbf{d}$, $\tau = \Psi(M)$ and $\Psi : \mathbb{G} \rightarrow \mathbb{Z}_p$ is a collision-resistant hash function. ϱ is the ciphertext of random mask u under a malleable (and also rerandomizable) encryption scheme (see Sect. 3.1). At the high level, the strand C_1 carries the message while the strand C_2 is to help rerandomize C_1 without public key. The encrypted mask u shared between C_1 and C_2 disables the adversary to mix together strands from two different ciphertexts (of the same plaintext) to obtain a valid ciphertext. The exponents of strand C_1 are perturbed by an additional vector \mathbf{z} to restrict the manner of recombining the two strands. Consequently, to rerandomize ciphertext ζ , one randomly picks $v \in \overline{\mathbb{G}}$, $s, t \in \mathbb{Z}_p^*$, $\overline{s}, \overline{t} \in \mathbb{Z}_q^*$ and computes

$$C'_1 := ([v \cdot u(\mathbf{x} + \mathbf{z}) + sv \cdot u\mathbf{y}], M \cdot [\mathbf{b}^\top \mathbf{x}] \cdot [s\mathbf{b}^\top \mathbf{y}], [\boldsymbol{\alpha}^\top \mathbf{x}] \cdot [s\boldsymbol{\alpha}^\top \mathbf{y}]),$$

$$C'_3 := ([\overline{\mathbf{x}} + \overline{s} \cdot \overline{\mathbf{y}}], v \cdot u \cdot [\overline{\mathbf{b}}^\top \overline{\mathbf{x}}] \cdot [\overline{s}\overline{\mathbf{b}}^\top \overline{\mathbf{y}}], [\overline{\mathbf{c}}^\top \overline{\mathbf{x}}] \cdot [\overline{s}\overline{\mathbf{c}}^\top \overline{\mathbf{y}}]),$$

$$C'_2 := ([tv \cdot u\mathbf{y}], [t\mathbf{b}^\top \mathbf{y}], [t\boldsymbol{\alpha}^\top \mathbf{y}]) \text{ and } C'_4 := ([\overline{t} \cdot \overline{\mathbf{y}}], [\overline{t}\overline{\mathbf{b}}^\top \overline{\mathbf{y}}], [\overline{t}\overline{\mathbf{c}}^\top \overline{\mathbf{y}}]).$$

Partial rerandomizability breaking the receiver-anonymity. It is shown in [22] that the above is the only valid way for full rerandomization of ciphertext. However, one can note that strands C_3 and C_4 can also be rerandomized with public keys $[\overline{\mathbf{b}}^\top \overline{\mathbf{g}}]$ and $[\overline{\mathbf{c}}^\top \overline{\mathbf{g}}]$ as follows.

$$C'_3 := \left([\overline{\mathbf{x}} + \overline{s} \cdot \overline{\mathbf{g}}], u \cdot [\overline{\mathbf{b}}^\top \overline{\mathbf{x}}] \cdot [\overline{s}\overline{\mathbf{b}}^\top \overline{\mathbf{g}}], [\overline{\mathbf{c}}^\top \overline{\mathbf{x}}] \cdot [\overline{s}\overline{\mathbf{c}}^\top \overline{\mathbf{g}}] \right),$$

$$C'_4 := \left([\overline{\mathbf{y}} + \overline{t} \cdot \overline{\mathbf{g}}], [\overline{\mathbf{b}}^\top \overline{\mathbf{y}}] \cdot [\overline{t}\overline{\mathbf{b}}^\top \overline{\mathbf{g}}], [\overline{\mathbf{c}}^\top \overline{\mathbf{y}}] \cdot [\overline{t}\overline{\mathbf{c}}^\top \overline{\mathbf{g}}] \right),$$

where $\overline{s}, \overline{t} \in \mathbb{Z}_q^*$. We now demonstrate why the PR scheme is not RCCA receiver-anonymous. Recalling the game of RCCA receiver-anonymity in Fig. 2, the adversary has access to a guarded decryption oracle which on input ζ , first computes $M_0 = \text{Dec}(\text{SK}_0, \zeta)$ and $M_1 = \text{Dec}(\text{SK}_1, \zeta)$, then checks if $M \in \{M_0, M_1\}$. If so, it returns **replay**, otherwise it returns (M_0, M_1) . As for the PR scheme, adversary could obtain a ciphertext ζ_0^* by rerandomizing strands C_3 and C_4 in the challenge ciphertext ζ^* with public key PK_0 in the above way. If $b = 0$, ζ_0^* is a valid ciphertext of M ; otherwise, ζ_0^* is invalid. With the response of the guarded decryption oracle, the adversary is able to distinguish these two cases.

Our concrete treatment of the PR scheme for RCCA receiver-anonymity. To achieve RCCA receiver-anonymity, we have to disable the rerandomization of

strands C_3 and C_4 employing the public key. Note that the rerandomization of strands C_1 and C_2 is restricted by mask u and vector \mathbf{z} . If we also apply this technique to C_3 and C_4 , extra strands are required to encrypt the mask in C_3 and C_4 , which would incur the partial rerandomization of ciphertext employing the public key again. To bypass this problem, we move the masks and additional vectors to the validity checking components of strands. Since the validity checking part contains only one component, an additional component is appended to each strand for perturbation on the validity checking part. Concretely, the ciphertext of our variant is:

$$\zeta := \left(\underbrace{[\mathbf{x}], M \cdot [\mathbf{b}^\top \mathbf{x}], [u\boldsymbol{\alpha}^\top \mathbf{x}^\dagger], [u\boldsymbol{\beta}^\top \mathbf{x}^\dagger]}_{C_1: \text{message-carrying strand}}, \underbrace{[\mathbf{y}], [\mathbf{b}^\top \mathbf{y}], [u\boldsymbol{\alpha}^\top \mathbf{y}], [u\boldsymbol{\beta}^\top \mathbf{y}]}_{C_2: \text{rerandomization strand}}, \varrho \right),$$

$$\varrho := \left(\underbrace{[\bar{\mathbf{x}}], u \cdot [\bar{\mathbf{b}}^\top \bar{\mathbf{x}}], [u\bar{\mathbf{c}}^\top \bar{\mathbf{x}}^\dagger], [u\bar{\mathbf{d}}^\top \bar{\mathbf{x}}^\dagger]}_{C_3: \text{mask-carrying strand}}, \underbrace{[\bar{\mathbf{y}}], [\bar{\mathbf{b}}^\top \bar{\mathbf{y}}], [u\bar{\mathbf{c}}^\top \bar{\mathbf{y}}], [u\bar{\mathbf{d}}^\top \bar{\mathbf{y}}]}_{C_4: \text{rerandomization strand}} \right) \tag{2}$$

where $u \in \overline{\mathbb{G}}$, $\mathbf{x}^\dagger = \mathbf{x} + z_1 \mathbf{g}$, $\mathbf{x}^\ddagger = \mathbf{x} + z_2 \mathbf{g}$ for $z_1, z_2 \in \mathbb{Z}_p^*$ with $z_1 \neq z_2$, $\bar{\mathbf{x}}^\dagger = \bar{\mathbf{x}} + \bar{z}_1 \bar{\mathbf{g}}$, $\bar{\mathbf{x}}^\ddagger = \bar{\mathbf{x}} + \bar{z}_2 \bar{\mathbf{g}}$ for $\bar{z}_1, \bar{z}_2 \in \mathbb{Z}_q^*$ with $\bar{z}_1 \neq \bar{z}_2$, $\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f} \in \mathbb{Z}_p^2$, $\boldsymbol{\alpha} = \mathbf{c} + m\mathbf{d}$, $\boldsymbol{\beta} = \mathbf{e} + m\mathbf{f}$, $m = \Psi(M)$ and $\Psi : \mathbb{G} \rightarrow \mathbb{Z}_p$ is a collision-resistant hash function. The rerandomization of strands C_1, C_2 is still restricted by mask u and vector (z_1, z_2) . As for strands C_3, C_4 , their rerandomization can be restricted by mask u and vector (\bar{z}_1, \bar{z}_2) , since u is placed on validity checking part.

We stress that the above modifications are carefully conducted to preserve the RCCA security of the encryption scheme. First of all, extra secret keys (e.g., \mathbf{e}, \mathbf{f} and $\bar{\mathbf{d}}$) are introduced to compute the additional component in validity checking part such that, given a valid ciphertext ζ , the attacker cannot infer a new validity checking part for particular $[\mathbf{x}]$ or $[\bar{\mathbf{x}}]$ (that cannot be obtained by re-encrypting ζ). Secondly, the usage of mask u in strands C_3, C_4 is safe and sound. Taking component $[u\bar{\mathbf{c}}^\top \bar{\mathbf{x}}^\dagger]$ as example, it is equivalent to the value of $[(u \bmod q)\bar{\mathbf{c}}^\top \bar{\mathbf{x}}^\dagger]$, as mask u is an integer in \mathbb{Z}_p^* . Since the modular operation satisfies the homomorphism property, the re-encryption on strands C_3, C_4 maintains correctness. Note that a component in the validity checking part actually corresponds to two different masks u, u' with $u' = u \bmod q$. We remark that this would not affect the RCCA security as long as the size of the modulus q is large enough so that the attacker cannot guess the value of mask u trivially.

Generalization of our approach. Note that the ciphertext structure of our above variant still shares some similarities with that of the PR scheme which is essentially a double “strand” of Cramer-Shoup ciphertext. We turn to explore whether it is possible to generalize our treatment following the CS-paradigm [8].

We start by recalling the CS-paradigm based on SPHF, and then seek to extend the notion of SPHF to interpret our proposed variant and its security.

Recalling Cramer-Shoup paradigm from SPHFs. Smooth Projective Hash Function (SPHF) was originally proposed by Cramer and Shoup [8] for generally constructing practical CCA-secure PKE. Roughly, SPHF is a family of hash

functions $\mathcal{H} = (H_{\text{sk}})_{\text{sk} \in \mathcal{K}}$ indexed by \mathcal{K} that map the non-empty element set \mathcal{X} onto the hash value set Π . Each SPHF is associated with an NP-language $\mathcal{L} \subset \mathcal{X}$ where elements in \mathcal{L} are computationally indistinguishable from those in $\mathcal{X} \setminus \mathcal{L}$ (i.e., hard subset membership problem). For any $x \in \mathcal{L}$, $H_{\text{sk}}(x)$ could be efficiently computed using either the hashing key $\text{sk} \in \mathcal{K}$, i.e., $\text{Priv}(\text{sk}, x) = H_{\text{sk}}(x)$ (*private evaluation mode*), or the projection key $\text{pk} = \phi(\text{sk}) \in \mathcal{P}$ with the witness $w \in \mathcal{W}$ to the fact $x \in \mathcal{L}$, i.e., $\text{Pub}(\text{pk}, x, w) = H_{\text{sk}}(x)$ (*public evaluation mode*). The notion of SPHF could be generalized to tag-based SPHF where a tag τ is also taken as an auxiliary input by $H_{(\cdot)}$, Priv and Pub . The CS-paradigm is based on a Smooth_1 SPHF $= (H_{(\cdot)}, \phi, \text{Priv}, \text{Pub})$ and a Smooth_2 tag-based $\widehat{\text{SPHF}} = (\widehat{H}_{(\cdot)}, \widehat{\phi}, \widehat{\text{Priv}}, \widehat{\text{Pub}})$. The public key is $(\text{pk}, \widehat{\text{pk}}) = (\phi(\text{sk}), \widehat{\phi}(\widehat{\text{sk}}))$ and the ciphertext is

$$\zeta := \left(x, M \cdot \text{Pub}(\text{pk}, x, w), \widehat{\text{Pub}}(\widehat{\text{pk}}, x, w, \tau) \right) = \left(x, M \cdot H_{\text{sk}}(x), \widehat{H}_{\widehat{\text{sk}}}(x, \tau) \right),$$

where $x \in \mathcal{L}$, w is the witness of x , $\tau = \Psi(x, M \cdot H_{\text{sk}}(x))$ and Ψ is a collision-resistant hash function. To make our later argument easier to follow, below we first provide an overview of justification of CCA security from SPHF. Consider the challenge ciphertext $\zeta^* = (x^*, M_b \cdot \pi^*, \widehat{\pi}^*)$ in the CCA security game.

- 1) Due to the hard subset membership problem, we can replace $x^* \in \mathcal{L}$ in ζ^* with $x^* \in \mathcal{X} \setminus \mathcal{L}$ and compute $\pi^* = \text{Priv}(\text{sk}, x^*)$, $\widehat{\pi}^* = \widehat{\text{Priv}}(\widehat{\text{sk}}, x^*, \tau^*)$.
- 2) By the Smooth_2 property of tag-based $\widehat{\text{SPHF}}$, any “bad” ciphertext ζ including $x \neq x^* \in \mathcal{X} \setminus \mathcal{L}$ will be rejected by the decryption oracle as $\widehat{\pi} = \widehat{H}_{\widehat{\text{sk}}}(x, \tau)$ is uniformly distributed, even conditioned on $\widehat{\text{pk}}$ and $\widehat{\pi}^*$.
- 3) By the Smooth_1 property of SPHF, π^* in ζ^* is uniformly distributed and thus ζ^* perfectly hides M_b , which yields the CCA security.

Generalization of our construction via newly extended SPHFs. As the first attempt to generalize our variant, we abstract strands C_1 and C_2 in Eq. (2) using the following SPHFs:

$$\text{SPHF} = (H_{(\cdot)}, \phi, \text{Priv}, \text{Pub}), \widehat{\text{SPHF}} = (\widehat{H}_{(\cdot)}, \widehat{\phi}, \widehat{\text{Priv}}, \widehat{\text{Pub}}), \widetilde{\text{SPHF}} = (\widetilde{H}_{(\cdot)}, \widetilde{\phi}, \widetilde{\text{Priv}}, \widetilde{\text{Pub}}),$$

based on which C_1 and C_2 in our variant could be written as

$$C_1 := \left([\mathbf{x}], M \cdot H_{\text{sk}}([\mathbf{x}]), \left[\widehat{H}_{\widehat{\text{sk}}}([\mathbf{x}], \tau) \right] \right), \quad C_2 := \left([\mathbf{y}], H_{\text{sk}}([\mathbf{y}]), \left[\widetilde{H}_{\widetilde{\text{sk}}}([\mathbf{y}], \tau) \right] \right), \quad (3)$$

where tag $\tau = (u, m)$, hashing key $\widehat{\text{sk}} = (\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f})$ and $\widetilde{\text{sk}} = \widehat{\text{sk}}$. Note that these SPHFs are defined on the same set $\mathcal{X} = \{[\mathbf{a}] | \mathbf{a} \in \mathbb{Z}_p^2\}$ with NP-language $\mathcal{L} = \{[r\mathbf{g}] | r \in \mathbb{Z}_p\}$ for $\mathbf{g} \in \mathbb{Z}_p^2$. The rerandomization of C_1 and C_2 is defined as

$$C'_1 = \left([\mathbf{x} + \mathbf{s}\mathbf{y}], M \cdot \underbrace{\left[\mathbf{b}^\top \mathbf{x} \right] \cdot \left[\mathbf{s}\mathbf{b}^\top \mathbf{y} \right]}_{\underbrace{H_{\text{sk}}([\mathbf{x}]) \cdot (H_{\text{sk}}([\mathbf{y}]))^s}_{(H_{\text{sk}}([\mathbf{y}]))^t}}, \underbrace{\left[\widehat{H}_{\widehat{\text{sk}}}([\mathbf{x}], \tau) \right]^u \cdot \left[\widetilde{H}_{\widetilde{\text{sk}}}([\mathbf{y}], \tau) \right]^{sv}}_{\left(\widehat{H}_{\widehat{\text{sk}}}([\mathbf{y}], \tau) \right)^{tv}} \cdot \left[\mathbf{v}\mathbf{u}\boldsymbol{\alpha}^\top \mathbf{x}^\dagger \right] \cdot \left[\mathbf{s}\mathbf{v}\mathbf{u}\boldsymbol{\alpha}^\top \mathbf{y} \right], \left[\mathbf{v}\mathbf{u}\boldsymbol{\beta}^\top \mathbf{x}^\dagger \right] \cdot \left[\mathbf{s}\mathbf{v}\mathbf{u}\boldsymbol{\beta}^\top \mathbf{y} \right] \right)$$

$$C'_2 = \left([t\mathbf{y}], \left[\mathbf{t}\mathbf{b}^\top \mathbf{y} \right], \left[\mathbf{t}\mathbf{v} \cdot \mathbf{u}\boldsymbol{\alpha}^\top \mathbf{y} \right], \left[\mathbf{t}\mathbf{v} \cdot \mathbf{u}\boldsymbol{\beta}^\top \mathbf{y} \right] \right),$$

where $v \leftarrow_s \overline{\mathbb{G}}$, $s, t \leftarrow_s \mathbb{Z}_p^*$. The generalization of strand $C_3(C_4)$ is similar to that of $C_1(C_2)$ and can be denoted by SPHF's defined on the same set $\overline{\mathcal{X}} = \{[\overline{\mathbf{a}}] | \overline{\mathbf{a}} \in \mathbb{Z}_q^2\}$ with NP-language $\overline{\mathcal{L}} = \{[\overline{r\mathbf{g}}] | \overline{r} \in \mathbb{Z}_q\}$ for $\overline{\mathbf{g}} \in \mathbb{Z}_q^2$. The ciphertext rerandomization in our variant could be classified with respect to SPHF's as follows.

- *Self-rerandomization within same SPHF*, e.g.,

$$(H_{\text{sk}}([\mathbf{x}]), H_{\text{sk}}([\mathbf{y}])) \rightsquigarrow H_{\text{sk}}([\mathbf{x}]) \cdot (H_{\text{sk}}([\mathbf{y}]))^s$$

- *Pairwise-rerandomization between different SPHF's*, e.g.,

$$\left(\widehat{H}_{\widehat{\text{sk}}}([\mathbf{x}], \tau), \widetilde{H}_{\widetilde{\text{sk}}}([\mathbf{y}], \tau) \right) \rightsquigarrow \left(\widehat{H}_{\widehat{\text{sk}}}([\mathbf{x}], \tau) \right)^v \cdot \left(\widetilde{H}_{\widetilde{\text{sk}}}([\mathbf{y}], \tau) \right)^{sv}$$

Motivated by these observations, we put forward the notion of *rerandomizable* SPHF (Re-SPHF) which is a regular SPHF augmented with self- and pairwise-rerandomizability. Specifically, based on the typical definition of SPHF, we formalize three extra algorithms namely RandX, RandT and RandH to capture both cases of rerandomization. The correctness of ciphertext in our variant is guaranteed by the *rerandomization correctness* with respect to RandX, RandT and RandH in Re-SPHF, while the perfect rerandomization of ciphertext is captured by the notion of perfect rerandomization in Re-SPHF's.

Arguments of RCCA security with receiver-anonymity. Analogous to the classification of rerandomization, we redefine two types of smoothness for Re-SPHF as below. Let $\text{CRX}(x^*)$ denote the set of all rerandomization of x^* obtained via RandX, $\text{CRX}(x_1^*, x_2^*)$ denote the set of all rerandomization of x_1^* obtained via RandX with x_2^* and $\text{CRT}(\tau^*)$ denote the set of all rerandomization of τ^* obtained via RandT. Let $\stackrel{s}{\equiv}$ denote statistical indistinguishability between distributions.

- *Controlled-Self-Rerandomizable Smoothness (CSR-Smooth)*. For any $x^* \in \mathcal{X}$, $\tau^* \in \mathcal{T}$ and $(x, \tau) \in \mathcal{X} \setminus \mathcal{L} \times \mathcal{T}$ with $x \notin \text{CRX}(x^*)$ or $\tau \notin \text{CRT}(\tau^*)$,

$$\left(\text{pk}, H_{\text{sk}}(x^*, \tau^*), \boxed{H_{\text{sk}}(x, \tau)} \right) \stackrel{s}{\equiv} \left(\text{pk}, H_{\text{sk}}(x^*, \tau^*), \boxed{\pi} \leftarrow_s \Pi \right).$$

- *Controlled-Pairwise-Rerandomizable Smoothness (CPR-Smooth)*. For any $x_1^*, x_2^* \in \mathcal{X}$, $\tau^* \in \mathcal{T}$ and $(x, \tau) \in \mathcal{X} \setminus \mathcal{L} \times \mathcal{T}$ with $x \notin \text{CRX}(x_1^*, x_2^*)$ or $\tau \notin \text{CRT}(\tau^*)$,

$$\left(\widehat{\text{pk}}, \widehat{H}_{\widehat{\text{sk}}}(x_1^*, \tau^*), \widetilde{H}_{\widetilde{\text{sk}}}(x_2^*, \tau^*), \boxed{\widehat{H}_{\widehat{\text{sk}}}(x, \tau)} \right) \stackrel{s}{\equiv} \left(\widehat{\text{pk}}, \widehat{H}_{\widehat{\text{sk}}}(x_1^*, \tau^*), \widetilde{H}_{\widetilde{\text{sk}}}(x_2^*, \tau^*), \boxed{\pi} \leftarrow_s \widehat{\Pi} \right),$$

where $\widehat{\text{sk}} = \widetilde{\text{sk}}$. Also, we redefine two enhanced Smooth₁ for Re-SPHF as below.

- *Self-Twin 1-Smoothness (ST-Smooth₁)*. For $x_1, x_2 \leftarrow_s \mathcal{X} \setminus \mathcal{L}$ and $\tau \leftarrow_s \mathcal{T}$,

$$\left(\text{pk}, \boxed{H_{\text{sk}}(x_1, \tau)}, \boxed{H_{\text{sk}}(x_2, \tau)} \right) \stackrel{s}{\equiv} \left(\text{pk}, \boxed{\pi_1} \leftarrow_s \Pi, \boxed{\pi_2} \leftarrow_s \Pi \right).$$

- *Pairwise-Twin 1-Smoothness (PT-Smooth₁)*. For $x_1, x_2 \leftarrow_s \mathcal{X} \setminus \mathcal{L}$ and $\tau \leftarrow_s \mathcal{T}$,

$$\left(\widehat{\text{pk}}, \boxed{\widehat{H}_{\widehat{\text{sk}}}(x_1, \tau)}, \boxed{\widetilde{H}_{\widetilde{\text{sk}}}(x_2, \tau)} \right) \stackrel{s}{\equiv} \left(\widehat{\text{pk}}, \boxed{\pi_1} \leftarrow_s \widehat{\Pi}, \boxed{\pi_2} \leftarrow_s \widetilde{\Pi} \right).$$

We now show how to realize RCCA security and receiver-anonymity with these new properties. Consider a challenge ciphertext ζ^* with words $[\mathbf{x}^*], [\mathbf{y}^*] \in \mathcal{L}$ and $[\bar{\mathbf{x}}^*], [\bar{\mathbf{y}}^*] \in \bar{\mathcal{L}}$ in the RCCA security game. Similar to the security justification of CS-paradigm, below we provide the arguments to justify the RCCA security of our variant.

- 1) Due to the hard subset membership problems on $(\mathcal{X}, \mathcal{L})$ and $(\bar{\mathcal{X}}, \bar{\mathcal{L}})$, the challenge ciphertext ζ^* generated by alternative encryption algorithm, where $[\mathbf{x}^*], [\mathbf{y}^*] \in \mathcal{L}$ and $[\bar{\mathbf{x}}^*], [\bar{\mathbf{y}}^*] \in \bar{\mathcal{L}}$ are replaced with non-words (i.e., $[\mathbf{x}^*], [\mathbf{y}^*] \in \mathcal{X} \setminus \mathcal{L}$ and $[\bar{\mathbf{x}}^*], [\bar{\mathbf{y}}^*] \in \bar{\mathcal{X}} \setminus \bar{\mathcal{L}}$) and the corresponding hash values are computed with hashing keys, is computationally indistinguishable from one generated by original encryption algorithm.
- 2) Note that the Smooth_2 property used for proving the CS-paradigm is not satisfied here as the adversary may construct a valid ciphertext with at least one non-word via rerandomizing ζ^* . Fortunately, the manner to rerandomize ζ^* in our variant is restricted by $z_1, z_2, \bar{z}_1, \bar{z}_2, u$ and querying such a “valid” rerandomization of ζ^* will not leak information about private key. To the end, a computationally unbounded decryption oracle with public key and challenge ciphertext ζ^* only will reject “bad” ciphertext ζ that includes at least one non-word but is not a “valid” rerandomization of ζ^* , as the corresponding hash values (e.g., $\tilde{H}_{\widehat{\tau}}([\mathbf{y}], \tau)$ and $\widehat{H}_{\widehat{\tau}}([\mathbf{x}], \tau)$) in ciphertext ζ are uniformly distributed by properties CSR-Smooth and CPR-Smooth.
- 3) By properties ST-Smooth₁ and PT-Smooth₁, all the hash values in ζ^* are uniformly distributed conditioned on public key, and M_b is perfectly hidden in ζ^* , which yields the RCCA security of our variant.

Note that RCCA security guarantees the privacy of the underlying plaintext, while RCCA receiver-anonymity captures the privacy of the public key. The justification for receiver-anonymity is indeed similar to the above arguments. In particular, the decryption oracle also relies on CSR-Smooth and CPR-Smooth properties to reject all the “bad” ciphertexts. In the end, the uniform distributions of all the hash values in ζ^* imply the receiver-anonymity in RCCA setting.

Related Work. Here we illustrate several previous constructions of Rand-RCCA-secure PKE and provide an efficiency comparison with our scheme, putting aside the receiver-anonymity. Also, some related SPHF’s variants will be given.

Non-anonymous constructions. Groth [14] presented a perfect Rand-RCCA-secure scheme, where the ciphertext can be rerandomized into another one in an unlinkable way, under the generic group model, and the ciphertext size expansion is as large as the bit-length of the plaintext. Phan and Pointcheval [21] then designed an efficient framework of RCCA-secure scheme, while Faonio and Fiore [10] showed that the rerandomizability of its ElGamal-based instantiation in [20] cannot resist any active attacks. Chase et al. [6] introduced a new way to construct perfect Rand-RCCA-secure PKE from a malleable NIZK system, where their construction has public verifiability property. Libert et al. [17] proposed a new construction that improves on Chase et al.’s scheme but still suffers from high computational costs and large ciphertext size (of 62 group elements) due

Table 1. Comparison of Rand-RCCA-secure PKE schemes ($k=2$). $|\text{PK}|$ and $|\text{CT}|$ represent the number of elements in public key and ciphertext, where ℓ denotes the bit-length of plaintext. Here \mathbb{G} and $\overline{\mathbb{G}}$ are standard DDH groups that satisfy certain requirements. $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are groups in bilinear pairing. Here $E, \overline{E}, E_1, E_2, E_T$ denote the execution time of exponentiation on $\mathbb{G}, \overline{\mathbb{G}}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the time cost of pairing is P . “Std” refers to standard model, “GGM” refers to generic group model, and “NPR” refers to non-programmable random oracle model. “Perfect” indicates perfect rerandomizability, “Universal” indicates that ciphertext rerandomization does not require the public key, and “Anonymity” refers to RCCA receiver-anonymity.

PKE	Groth04 [14]	PR07 [22]	LPQ17 [17]	FFHR19 [11]	FF20[10]	Ours (k -LIN)
$ \text{PK} $	$O(\ell)\mathbb{G}$	$4\overline{\mathbb{G}} + 7\mathbb{G}$	$11\mathbb{G}_1 + 16\mathbb{G}_2$	$7\mathbb{G}_1 + 7\mathbb{G}_2 + 2\mathbb{G}_T$	$11\mathbb{G}$	$6\overline{\mathbb{G}} + 10\mathbb{G}$
$ \text{CT} $	$O(\ell)\mathbb{G}$	$8\overline{\mathbb{G}} + 12\mathbb{G}$	$42\mathbb{G}_1 + 20\mathbb{G}_2$	$3\mathbb{G}_1 + 2\mathbb{G}_2 + \mathbb{G}_T$	$11\mathbb{G}$	$12\overline{\mathbb{G}} + 12\mathbb{G}$
Enc	$O(\ell)E$	$8\overline{E} + 14E$	$79E_1 + 64E_2$	$4E_1 + 5E_2 + 3E_T + 5P$	$15E$	$12\overline{E} + 16E$
Dec	$O(\ell)E$	$8\overline{E} + 24E$	$1E_1 + 142P$	$8E_1 + 4E_2 + 4P$	$18E$	$18\overline{E} + 18E$
Rerand	$O(\ell)E$	$8\overline{E} + 16E$	$48E_1 + 24E_2$	$6E_1 + 7E_2 + 3E_T + 9P$	$11E$	$14\overline{E} + 14E$
Model	GGM	Std	Std	Std	NPR	Std
Assumption	DDH	DDH	SXDH	\mathcal{D}_k -MDDH	DDH	k -Linear
Perfect	✓	✓	✓	✓	×	✓
Universal	×	✓	×	×	×	✓
Anonymity	×	×	×	×	×	✓

to the adoption of NIZK. Recently, Faonio et al. [11] gave a new construction of perfect Rand-RCCA-secure PKE from \mathcal{D}_k -MDDH assumption. The ciphertext in their scheme (when $k=1$) is extremely short and consists of only 6 group elements. In a most recent work, Faonio and Fiore [10] proposed a more efficient Rand-RCCA-secure PKE with only weak rerandomizability, and where security is justified in the random oracle model.

In Table 1, we compare our scheme with previous works, putting aside our exclusive property of receiver-anonymity. Compared with the recent work of Faonio et al. [11], our 2-LIN-based instantiation, although based on special groups which are larger than a regular setting, does not involve any pairing computations. *SPHF variants.* Variants of SPHF with new properties have also been proposed in the literature [4, 7, 11, 15, 27]. Here we briefly introduce two works that are closely related to our Re-SPHF. Wee [27] built the frameworks for constructing PKE satisfying key-dependent message (KDM) security using SPHF with homomorphic hash function. Faonio et al. [11] presented controlled-malleable smooth-projective hash function (cmSPHF), an extension of malleable smooth-projective hash function (mSPHF) by Chen et al. in [7] with respect to elements and tags. However, the cmSPHF cannot support universal rerandomizability.

3 Preliminaries

Let $n \in \mathbb{N}$ denote the security parameter and $\text{negl}(\cdot)$ denote the negligible function. For $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{Z}_p^n$ and $g \in \mathbb{G}$, $[\mathbf{x}]$ denotes vector $(g^{x_1}, \dots, g^{x_n})$. For set \mathcal{X} , $x \leftarrow_s \mathcal{X}$ denotes that x is sampled uniformly from \mathcal{X} at random. For any randomized algorithm \mathcal{F} , $y \leftarrow_s \mathcal{F}(x)$ denotes the random output of \mathcal{F} .

3.1 Public-Key Encryption (PKE)

A PKE scheme consists of algorithms $(\text{KGen}, \text{Enc}, \text{Dec})$: $\text{KGen}(1^n)$ takes as input the security parameter 1^n , and outputs the key pair (PK, SK) ; The encryption algorithm $\text{Enc}(\text{PK}, M)$ takes as input the public key PK and the plaintext M , and outputs the ciphertext ζ ; The decryption algorithm $\text{Dec}(\text{SK}, \zeta)$ takes as input the secret key SK and the ciphertext ζ , and outputs the plaintext M or \perp .

A PKE scheme should satisfy *decryption correctness* which captures the fact that, for $(\text{PK}, \text{SK}) \leftarrow_{\$} \text{KGen}(1^n)$, for any $M \in \mathcal{M}$ (in valid message space),

$$\Pr[\text{Dec}(\text{SK}, \zeta) \neq M : \zeta \leftarrow_{\$} \text{Enc}(\text{PK}, M)] \leq \text{negl}(n).$$

Below we provide the definitions of rerandomizable PKE. As mentioned above, in this work, we are mainly interested in “universal rerandomization” that does not require the public key, which is crucial to realize receiver-anonymity. Therefore, we mainly follow the definitions given in [22].

Rerandomizable PKE. We say a PKE scheme is (universally) *rerandomizable* if there exists algorithm Rerand that takes as input ciphertext ζ and outputs a new ciphertext ζ' ; and for $(\text{PK}, \text{SK}) \leftarrow_{\$} \text{KGen}(1^n)$, any (possibly malicious) ciphertext ζ ,

$$\Pr[\text{Dec}(\text{SK}, \zeta') \neq \text{Dec}(\text{SK}, \zeta) : \zeta' \leftarrow_{\$} \text{Rerand}(\zeta)] \leq \text{negl}(n).$$

Definition 1 (Perfectly Rerandomizable PKE [11]). Assume $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec}, \text{Rerand})$ is rerandomizable. We say PKE is perfectly rerandomizable if following properties are satisfied.

- For $(\text{PK}, \text{SK}) \leftarrow_{\$} \text{KGen}(1^n)$, any $M \in \mathcal{M}$ and any (honestly generated) ciphertext ζ in the support of $\text{Enc}(\text{PK}, M)$, the distribution of $\text{Rerand}(\zeta)$ is identical to that of $\text{Enc}(\text{PK}, M)$.
- For $(\text{PK}, \text{SK}) \leftarrow_{\$} \text{KGen}(1^n)$ and any (possibly unbounded) adversary \mathcal{A} , given PK , the probability of \mathcal{A} generating a ciphertext ζ such that $\text{Dec}(\text{SK}, \zeta) = M \neq \perp$ for some M and ζ is not in the range of $\text{Enc}(\text{PK}, M)$ is negligible.

Coupled with the second property, called the tightness of decryption in both [22] and [11], the first property can be extended to any malicious ciphertext that decrypts successfully.

Malleable PKE. We say a PKE scheme is *malleable* if there exists an algorithm Maul that takes as input a ciphertext ζ and a message M' , and outputs a new ciphertext ζ' ; and for $(\text{PK}, \text{SK}) \leftarrow_{\$} \text{KGen}(1^n)$, any $M, M' \in \mathcal{M}$ and $\zeta \leftarrow_{\$} \text{Enc}(\text{PK}, M)$,

$$\Pr[\text{Dec}(\text{SK}, \zeta') \neq M \cdot M' : \zeta' \leftarrow_{\$} \text{Maul}(\zeta, M')] \leq \text{negl}(n).$$

W.l.o.g., we assume that message space \mathcal{M} is a multiplicative group, and let “ \cdot ” denote multiplication operation on \mathcal{M} .

Security definitions. We follow the definitions of RCCA security and RCCA receiver-anonymity in [22].

$\text{IND-RCCA}_{\text{PKE}}^{\mathcal{A}}(n)$	$\mathcal{DO}_{\text{SK}}(\zeta)$
$(\text{PK}, \text{SK}) \leftarrow_{\$} \text{KGen}(1^n)$	return $\text{Dec}(\text{SK}, \zeta)$
$(M_0, M_1) \leftarrow \mathcal{A}^{\mathcal{DO}_{\text{SK}}}(\text{PK})$	$\mathcal{GD}_{\text{SK}}^{M_0, M_1}(\zeta)$
$b \leftarrow_{\$} \{0, 1\}$	<hr/>
$\zeta^* \leftarrow_{\$} \text{Enc}(\text{PK}, M_b)$	$M := \text{Dec}(\text{SK}, \zeta)$
$b' \leftarrow \mathcal{A}^{\mathcal{GD}_{\text{SK}}^{M_0, M_1}}(\text{PK}, \zeta^*)$	if $M \in \{M_0, M_1\}$, return replay
if $b = b'$, return 1	else return M
else return 0	

Fig. 1. Definition of IND-RCCA game.

$\text{ANON-RCCA}_{\text{PKE}}^{\mathcal{A}}(n)$	$\mathcal{DO}_{\text{SK}_0, \text{SK}_1}(\zeta)$
$(\text{PK}_0, \text{SK}_0) \leftarrow_{\$} \text{KGen}(1^n)$	return $(\text{Dec}(\text{SK}_0, \zeta), \text{Dec}(\text{SK}_1, \zeta))$
$(\text{PK}_1, \text{SK}_1) \leftarrow_{\$} \text{KGen}(1^n)$	$\mathcal{GD}_{\text{SK}_0, \text{SK}_1}^M(\zeta)$
$M \leftarrow \mathcal{A}^{\mathcal{DO}_{\text{SK}_0, \text{SK}_1}}(\text{PK}_0, \text{PK}_1)$	<hr/>
$b \leftarrow_{\$} \{0, 1\}$	$M_0 := \text{Dec}(\text{SK}_0, \zeta); M_1 := \text{Dec}(\text{SK}_1, \zeta)$
$\zeta^* \leftarrow_{\$} \text{Enc}(\text{PK}_b, M)$	if $M \in \{M_0, M_1\}$, return replay
$b' \leftarrow \mathcal{A}^{\mathcal{GD}_{\text{SK}_0, \text{SK}_1}^M}(\text{PK}_0, \text{PK}_1, \zeta^*)$	else return (M_0, M_1)
if $b = b'$, return 1	
else return 0	

Fig. 2. Definition of ANON-RCCA game.

Definition 2 (RCCA Security). Let $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec})$ be a PKE scheme. Consider the security game $\text{IND-RCCA}_{\text{PKE}}^{\mathcal{A}}(n)$ in Fig. 1. We say PKE is RCCA-secure if for any PPT algorithm \mathcal{A} in game $\text{IND-RCCA}_{\text{PKE}}^{\mathcal{A}}(n)$,

$$\text{Adv}_{\mathcal{A}, \text{PKE}}^{\text{IND-RCCA}}(n) := \left| \Pr \left[\text{IND-RCCA}_{\text{PKE}}^{\mathcal{A}}(n) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(n).$$

Definition 3 (RCCA Receiver-Anonymity). Let $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec})$ be a PKE scheme. Consider the security game $\text{ANON-RCCA}_{\text{PKE}}^{\mathcal{A}}(n)$ in Fig. 2. We say PKE is RCCA receiver-anonymous if for any PPT algorithm \mathcal{A} in game $\text{ANON-RCCA}_{\text{PKE}}^{\mathcal{A}}(n)$,

$$\text{Adv}_{\mathcal{A}, \text{PKE}}^{\text{ANON-RCCA}}(n) := \left| \Pr \left[\text{ANON-RCCA}_{\text{PKE}}^{\mathcal{A}}(n) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(n).$$

3.2 Smooth Projective Hash Function (SPHF)

In this work, we focus on a more general version of smooth projective hash function, called tag-based smooth projective hash function (tag-SPHF) [8]. The regular SPHF can be regarded as a special case of tag-SPHF with empty tag space $\mathcal{T} = \emptyset$. A tag-SPHF is associated with set \mathcal{X} , NP-language \mathcal{L} where $\mathcal{L} \subset \mathcal{X}$, and defined by four algorithms (Setup, ϕ , Priv, Pub) as follows:

- Setup(1^n) takes as input a security parameter 1^n , and outputs public parameters $\text{pp} = (\mathcal{K}, \mathcal{T}, \Pi, H_{(\cdot)})$, where \mathcal{K} is the hashing key space, \mathcal{T} is the tag space, Π is the hash value space, $H_{(\cdot)} : \mathcal{X} \times \mathcal{T} \rightarrow \Pi$ is an efficiently computable hash function family indexed by hashing key $\text{sk} \in \mathcal{K}$.
- $\phi(\text{sk})$ derives the projection key pk from the hashing key $\text{sk} \in \mathcal{K}$.
- Priv(sk, x, τ) takes as input an element $x \in \mathcal{X}$, tag $\tau \in \mathcal{T}$ and hashing key sk , and outputs hash value $\pi = H_{\text{sk}}(x, \tau) \in \Pi$.
- Pub(pk, x, w, τ) takes as input a word $x \in \mathcal{L}$ with witness w , tag τ and projection key pk , and outputs hash value $\pi = H_{\text{sk}}(x, \tau) \in \Pi$.

In regular SPHF, both the input of algorithms Priv(sk, x) and Pub(pk, x, w) do not include tag τ , and the outputted hash value is $\pi = H_{\text{sk}}(x)$.

Definition 4 (Correctness). For $\text{pp} \leftarrow_{\$} \text{Setup}(1^n)$, $\text{sk} \leftarrow_{\$} \mathcal{K}$ and $\text{pk} = \phi(\text{sk})$, any $x \in \mathcal{L}$ with witness w to the fact of $x \in \mathcal{L}$ and any $\tau \in \mathcal{T}$,

$$\Pr[\text{Priv}(\text{sk}, x, \tau) \neq \text{Pub}(\text{pk}, x, w, \tau)] \leq \text{negl}(n).$$

Assume that SPHF = (Setup, ϕ , Priv, Pub) is associated with \mathcal{X} , \mathcal{L} and \mathcal{T} .

Definition 5 (1-Smoothness). We say SPHF is Smooth₁ if for $\text{pp} \leftarrow_{\$} \text{Setup}(1^n)$, $\text{sk} \leftarrow_{\$} \mathcal{K}$, $\text{pk} = \phi(\text{sk})$ and any $(x, \tau) \in \mathcal{X} \setminus \mathcal{L} \times \mathcal{T}$, the following two distributions are statistically indistinguishable:

$$V_1 = \{(\text{pk}, x, \tau, \pi) \mid \pi = H_{\text{sk}}(x, \tau)\}, \quad V_2 = \{(\text{pk}, x, \tau, \pi') \mid \pi' \leftarrow_{\$} \Pi\}.$$

For certain tag-SPHFs, the smoothness property may be enhanced as follows.

Definition 6 (2-Smoothness). We say SPHF is Smooth₂ if for $\text{pp} \leftarrow_{\$} \text{Setup}(1^n)$, $\text{sk} \leftarrow_{\$} \mathcal{K}$, $\text{pk} = \phi(\text{sk})$, any $(x^*, \tau^*) \in \mathcal{X} \times \mathcal{T}$ and any $(x, \tau) \in \mathcal{X} \setminus \mathcal{L} \times \mathcal{T}$ with $(x, \tau) \neq (x^*, \tau^*)$, the following two distributions are statistically indistinguishable:

$$V_1 = \{(\text{pk}, x^*, \tau^*, x, \tau, H_{\text{sk}}(x^*, \tau^*), \pi) \mid \pi = H_{\text{sk}}(x, \tau)\},$$

$$V_2 = \{(\text{pk}, x^*, \tau^*, x, \tau, H_{\text{sk}}(x^*, \tau^*), \pi') \mid \pi' \leftarrow_{\$} \Pi\}.$$

We assume that it is efficient to sample elements from set \mathcal{X} and \mathcal{L} . Below we define the hard subset membership problem (SMP) between \mathcal{X} and \mathcal{L} .

Definition 7 (Hard Subset Membership Problem). We say the subset membership problem is hard on $(\mathcal{X}, \mathcal{L})$ if for any PPT adversary \mathcal{A} ,

$$|\Pr[\mathcal{A}(x) = 1] - \Pr[\mathcal{A}(x') = 1]| \leq \text{negl}(n),$$

where $x \leftarrow_{\$} \mathcal{L}$ and $x' \leftarrow_{\$} \mathcal{X}$.

4 Rerandomizable Tag-SPHF

4.1 Syntax of Rerandomizable Tag-SPHF

We slightly extend the typical SPHF syntax in such a way that the hash function family H is indexed not only by the hashing key $\text{sk} \in \mathcal{K}$ (as the typical case) but also by some (possible) auxiliary information ax , which is fixed as part of the public parameter. For generality and simplicity considerations, hereafter we assume that such information is public and implicitly included in the description of hash function family, and remain to use $H_{(\cdot)}$ instead of $H_{\text{ax},(\cdot)}$. Note that ax is set as “null” for typical SPHFs. We remark that since now the hash function family is not solely indexed by the hash key, for two SPHFs that are even with the same $(\mathcal{X}, \mathcal{L}, \mathcal{K}, \mathcal{T}, \Pi)$, their corresponding hash function families are not necessarily the same due to the possibly different auxiliary index ax .

Definition 8 (Rerandomizable Tag-SPHF (Re-T-SPHF)). *Let I and I' be two tag-SPHFs associated with same sets \mathcal{X} and \mathcal{L} , sharing partially the same public parameter $(\mathcal{K}, \mathcal{T}, \Pi)$ but having (possibly) different hash function families $H_{(\cdot)}$ and $H'_{(\cdot)}$. We say I is **pairwise-rerandomizable** with respect to I' if:*

- There exist three efficient algorithms as below.
 - $I.\text{RandX}(x, x', r_x)$ takes as input elements $x, x' \in \mathcal{X}$ and randomness $r_x \in \mathcal{R}_x$, outputs a new element $x^* \in \mathcal{X}$;
 - $I.\text{RandT}(\tau, r_\tau)$ takes as input tag $\tau \in \mathcal{T}$ and randomness $r_\tau \in \mathcal{R}_\tau$, outputs a new tag $\tau^* \in \mathcal{T}$;
 - $I.\text{RandH}(\pi, \pi', r_x, r_\tau)$ takes as input hash values $\pi, \pi' \in \Pi$ and randomnesses $r_x \in \mathcal{R}_x, r_\tau \in \mathcal{R}_\tau$, outputs a rerandomized hash value $\pi^* \in \Pi$, where \mathcal{R}_x and \mathcal{R}_τ are randomness space for element and tag respectively.
- For $\text{sk} \leftarrow_{\$} \mathcal{K}$, any $x, x' \in \mathcal{X}$, any $\tau \in \mathcal{T}$, let $\pi = H_{\text{sk}}(x, \tau)$ and $\pi' = H'_{\text{sk}}(x', \tau)$,

$$\Pr \left[\begin{array}{l} r_x \leftarrow_{\$} \mathcal{R}_x; r_\tau \leftarrow_{\$} \mathcal{R}_\tau \\ x^* := I.\text{RandX}(x, x', r_x); \\ \tau^* := I.\text{RandT}(\tau, r_\tau) \\ \pi^* := I.\text{RandH}(\pi, \pi', r_x, r_\tau) \end{array} : H_{\text{sk}}(x^*, \tau^*) \neq \pi^* \right] \leq \text{negl}(n).$$

If $I' = I^1$, we say that I is **self-rerandomizable**. In this case, the input x and x' for algorithm RandX could be the same element. We say that I is **linearly rerandomizable** if for any $\pi, \pi', \Delta \in \Pi$ (w.l.o.g., considering Π as a multiplicative group), $r_x \leftarrow_{\$} \mathcal{R}_x, r_\tau \leftarrow_{\$} \mathcal{R}_\tau$, $I.\text{RandH}(\pi \cdot \Delta, \pi', r_x, r_\tau) = I.\text{RandH}(\pi, \pi', r_x, r_\tau) \cdot \Delta$.

Remark (Re-SPHF). For a regular rerandomizable SPHF (hereafter referred to as Re-SPHF) where tag space $\mathcal{T} = \emptyset$, the algorithm RandT is absent and the parameter r_τ in the input of algorithm RandH is explicitly omitted.

¹ That is, $H_{(\cdot)}$ and $H'_{(\cdot)}$ have the same auxiliary index (which could be “null”), and thus are the same (since they work on the same \mathcal{K}).

Definition 9 (Perfect Re-T-SPHF). Assume I is pairwise-rerandomizable with respect to I' . We say that I is **perfectly rerandomizable** on \mathcal{T}_s with respect to I' if for $\text{sk} \leftarrow_s \mathcal{K}$, any $x, x' \in \mathcal{X}$, any $\tau \in \mathcal{T}_s \subseteq \mathcal{T}$, $r_x \leftarrow_s \mathcal{R}_x$, $r_\tau \leftarrow_s \mathcal{R}_\tau$ and $\pi = H_{\text{sk}}(x, \tau)$, $\pi' = H'_{\text{sk}}(x', \tau)$, the following distributions are identical:

$$V_1 = \{(x'', \tau'', \pi'') \mid x'' \leftarrow_s \mathcal{X}; \tau'' \leftarrow_s \mathcal{T}_s; \pi'' = H_{\text{sk}}(x'', \tau'')\},$$

$$V_2 = \left\{ (x^*, \tau^*, \pi^*) \mid \begin{array}{l} x^* := I.\text{RandX}(x, x', r_x); \tau^* := I.\text{RandT}(\tau, r_\tau) \\ \pi^* := I.\text{RandH}(\pi, \pi', r_x, r_\tau) \end{array} \right\}.$$

If $\mathcal{T}_s = \mathcal{T}$, we say I is **perfectly pairwise-rerandomizable** with respect to I' . If $I' = I$, we say I is **perfectly self-rerandomizable** on \mathcal{T}_s .

4.2 Redefining Smoothness for Re-T-SPHFs

We define the property of smoothness for Re-T-SPHFs as below.

Definition 10 (Controlled-Self-Rerandomizable Smoothness). Let I be self-rerandomizable. Assume it is associated with sets \mathcal{X} and \mathcal{L} , and the public parameter is $(\mathcal{K}, \mathcal{T}, \Pi, H_{(\cdot)})$. Denote $\text{CRX}(x) = \{I.\text{RandX}(x, x, r_x) \mid r_x \in \mathcal{R}_x\}$ and $\text{CRT}(\tau) = \{I.\text{RandT}(\tau, r_\tau) \mid r_\tau \in \mathcal{R}_\tau\}$. We say I satisfies **controlled-self-rerandomizable smoothness** (CSR-Smooth) if for $\text{sk} \leftarrow_s \mathcal{K}$ and $\text{pk} := I.\phi(\text{sk})$, any $(x^*, \tau^*) \in \mathcal{X} \times \mathcal{T}$ and any $(x, \tau) \in \mathcal{X} \setminus \mathcal{L} \times \mathcal{T}$ with $x \notin \text{CRX}(x^*)$ or $\tau \notin \text{CRT}(\tau^*)$, the following two distributions are statistically indistinguishable,

$$V_1 = \{(\text{pk}, x^*, x, H_{\text{sk}}(x^*, \tau^*), \pi) \mid \pi = H_{\text{sk}}(x, \tau)\},$$

$$V_2 = \{(\text{pk}, x^*, x, H_{\text{sk}}(x^*, \tau^*), \pi') \mid \pi' \leftarrow_s \Pi\}.$$

Definition 11 (Controlled-Pairwise-Rerandomizable Smoothness). Let I be pairwise-rerandomizable with respect to I' . Assume they are associated with sets \mathcal{X} and \mathcal{L} , and work on $(\mathcal{K}, \mathcal{T}, \Pi)$. Let $H_{(\cdot)}$ and $H'_{(\cdot)}$ be the hash function family of I and I' respectively. Denote $\text{CRX}(x, x') = \{I.\text{RandX}(x, x', r_x) \mid r_x \in \mathcal{R}_x\}$ and $\text{CRT}(\tau) = \{I.\text{RandT}(\tau, r_\tau) \mid r_\tau \in \mathcal{R}_\tau\}$. We say I satisfies **controlled-pairwise-rerandomizable smoothness** (CPR-Smooth) with respect to I' if for $\text{sk} \leftarrow_s \mathcal{K}$ and $\text{pk} := I.\phi(\text{sk})$, any $(x_1^*, \tau_1^*), (x_2^*, \tau_2^*) \in \mathcal{X} \times \mathcal{T}$ with $\tau_1^* = \tau_2^*$ and any $(x, \tau) \in \mathcal{X} \setminus \mathcal{L} \times \mathcal{T}$ with $x \notin \text{CRX}(x_1^*, x_2^*)$ or $\tau \notin \text{CRT}(\tau_1^*)$, the following two distributions are statistically indistinguishable:

$$V_1 = \{(\text{pk}, x_1^*, x_2^*, x, H_{\text{sk}}(x_1^*, \tau_1^*), H'_{\text{sk}}(x_2^*, \tau_2^*), \pi) \mid \pi = H_{\text{sk}}(x, \tau)\},$$

$$V_2 = \{(\text{pk}, x_1^*, x_2^*, x, H_{\text{sk}}(x_1^*, \tau_1^*), H'_{\text{sk}}(x_2^*, \tau_2^*), \pi') \mid \pi' \leftarrow_s \Pi\}.$$

Definition 12 (Self-Twin 1-Smoothness). Let I be self-rerandomizable. Assume it is associated with sets \mathcal{X} and \mathcal{L} , and the public parameter is $(\mathcal{K}, \mathcal{T}, \Pi, H_{(\cdot)})$. We say I satisfies **self-twin 1-smoothness** (ST-Smooth₁) if for $\text{sk} \leftarrow_s \mathcal{K}$ and $\text{pk} := I.\phi(\text{sk})$, $x^*, x \leftarrow_s \mathcal{X} \setminus \mathcal{L}$, $\tau \leftarrow_s \mathcal{T}$, the following two distributions are statistically indistinguishable:

$$V_1 = \{(\text{pk}, x^*, x, \tau, \pi^*, \pi) \mid \pi^* = H_{\text{sk}}(x^*, \tau), \pi = H_{\text{sk}}(x, \tau)\},$$

$$V_2 = \{(\text{pk}, x^*, x, \tau, \pi'', \pi') \mid \pi'', \pi' \leftarrow_s \Pi\}.$$

Definition 13 (Pairwise-Twin 1-Smoothness). Let I be pairwise-rerandomizable with respect to I' . Assume they are associated with sets \mathcal{X} and \mathcal{L} , and work on $(\mathcal{K}, \mathcal{T}, \Pi)$. Let $H_{(\cdot)}$ and $H'_{(\cdot)}$ be the hash function family of I and I' respectively. We say I satisfies **pairwise-twin 1-smoothness** (PT-Smooth₁) with respect to I' if for $\text{sk} \leftarrow_s \mathcal{K}$ and $\text{pk} := I.\phi(\text{sk})$, $x^*, x \leftarrow_s \mathcal{X} \setminus \mathcal{L}$, $\tau \leftarrow_s \mathcal{T}$, the following two distributions are statistically indistinguishable:

$$\begin{aligned} V_1 &= \{(\text{pk}, x^*, x, \tau, \pi^*, \pi) \mid \pi^* = H_{\text{sk}}(x^*, \tau), \pi = H'_{\text{sk}}(x, \tau)\}, \\ V_2 &= \{(\text{pk}, x^*, x, \tau, \pi'', \pi') \mid \pi'', \pi' \leftarrow_s \Pi\}. \end{aligned}$$

5 A General Framework of Rand-RCCA-secure PKE

5.1 Our Generic Construction

The generic construction of the anonymous Rand-RCCA-secure scheme PKE = (KGen, Enc, Dec, Rerand) is depicted in Fig. 3 where the sub-scheme MPKE = (MKGen, MEnc, MDec, MRerand, Maul) is given in Fig. 4.

<p><u>KGen(1^n)</u></p> <p>$\text{sk}_0 \leftarrow_s \mathcal{K}_0$; $\text{sk}_1 \leftarrow_s \mathcal{K}_1$ $\text{pk}_0 := I_0.\phi(\text{sk}_0)$; $\text{pk}_1 := I_1.\phi(\text{sk}_1)$ $\text{sk}_2 := \text{sk}_1$; $\text{pk}_2 := \text{pk}_1$ $(\text{mpk}, \text{msk}) \leftarrow_s \text{MKGen}(1^n)$ $\text{SK} := (\text{sk}_0, \text{sk}_1, \text{sk}_2, \text{msk})$ $\text{PK} := (\text{pk}_0, \text{pk}_1, \text{pk}_2, \text{mpk})$ return (PK, SK)</p>	<p><u>Dec(SK, ζ)</u></p> <p>$u := \text{MDec}(\text{msk}, \varrho)$; if $u = \perp$, return \perp $\pi'_1 := I_0.\text{Priv}(\text{sk}_0, x_1)$; $\pi'_2 := I_0.\text{Priv}(\text{sk}_0, x_2)$ $M := e_1 \cdot \pi_1'^{-1}$; $\tau := (u, \psi(M))$ $\hat{\pi}'_1 := I_1.\text{Priv}(\text{sk}_1, x_1, \tau)$ $\tilde{\pi}'_2 := I_2.\text{Priv}(\text{sk}_2, x_2, \tau)$ if $(\hat{\pi}'_1, \tilde{\pi}'_2, \pi'_2) \neq (\hat{\pi}_1, \tilde{\pi}_2, \pi_2)$, return \perp else return M</p>
<p><u>Enc(PK, $M \in \Pi_0$)</u></p> <p>$x_1 \leftarrow_s \mathcal{L}$ with witness w_1 $x_2 \leftarrow_s \mathcal{L}$ with witness w_2 $u \leftarrow_s \overline{\Pi}_0$; $\tau := (u, \psi(M))$ $e_1 := I_0.\text{Pub}(\text{pk}_0, x_1, w_1) \cdot M$ $\hat{\pi}_1 := I_1.\text{Pub}(\text{pk}_1, x_1, w_1, \tau)$ $\pi_2 := I_0.\text{Pub}(\text{pk}_0, x_2, w_2)$ $\tilde{\pi}_2 := I_2.\text{Pub}(\text{pk}_2, x_2, w_2, \tau)$ $\varrho \leftarrow_s \text{MEnc}(\text{mpk}, u)$ return $\zeta := (x_1, e_1, \hat{\pi}_1, x_2, \pi_2, \tilde{\pi}_2, \varrho)$</p>	<p><u>Rerand(ζ)</u></p> <p>$r_1, r_2 \leftarrow_s \mathcal{R}_x$; $r_\tau \leftarrow_s \overline{\Pi}_0$ $x'_1 := I_0.\text{RandX}(x_1, x_2, r_1)$ $x'_2 := I_0.\text{RandX}(x_2, x_2, r_2)$ $e'_1 := I_0.\text{RandH}(e_1, \pi_2, r_1)$ $\hat{\pi}'_1 := I_1.\text{RandH}(\hat{\pi}_1, \tilde{\pi}_2, r_1, r_\tau)$ $\pi'_2 := I_0.\text{RandH}(\pi_2, \pi_2, r_2)$ $\tilde{\pi}'_2 := I_2.\text{RandH}(\tilde{\pi}_2, \tilde{\pi}_2, r_2, r_\tau)$ $\varrho' := \text{MRerand}(\text{Maul}(\varrho, r_\tau))$ return $\zeta' := (x'_1, e'_1, \hat{\pi}'_1, x'_2, \pi'_2, \tilde{\pi}'_2, \varrho')$</p>

Fig. 3. Our anonymous Rand-RCCA-secure scheme PKE

$\text{MKGen}(1^n)$ <hr/> $\begin{aligned} \overline{\text{sk}}_0 &\leftarrow_{\$} \overline{\mathcal{K}}_0; \text{sk}_3 \leftarrow_{\$} \mathcal{K}_3 \\ \overline{\text{pk}}_0 &:= \overline{I}_0.\phi(\overline{\text{sk}}_0) \\ \text{pk}_3 &:= I_3.\phi(\text{sk}_3) \\ \text{sk}_4 &:= \text{sk}_3; \text{pk}_4 := \text{pk}_3 \\ \text{msk} &:= (\overline{\text{sk}}_0, \text{sk}_3, \text{sk}_4) \\ \text{mpk} &:= (\overline{\text{pk}}_0, \text{pk}_3, \text{pk}_4) \\ \text{return} &(\text{mpk}, \text{msk}) \end{aligned}$ $\text{MEnc}(\text{mpk}, u \in \overline{\Pi}_0)$ <hr/> $\begin{aligned} x_3 &\leftarrow_{\$} \overline{\mathcal{L}} \text{ with witness } w_3 \\ x_4 &\leftarrow_{\$} \overline{\mathcal{L}} \text{ with witness } w_4 \\ \tau &:= u \\ e_3 &:= \overline{I}_0.\text{Pub}(\overline{\text{pk}}_0, x_3, w_3) \cdot u \\ \widehat{\pi}_3 &:= I_3.\text{Pub}(\text{pk}_3, x_3, w_3, \tau) \\ \pi_4 &:= \overline{I}_0.\text{Pub}(\overline{\text{pk}}_0, x_4, w_4) \\ \widetilde{\pi}_4 &:= I_4.\text{Pub}(\text{pk}_4, x_4, w_4, \tau) \\ \varrho &:= (x_3, e_3, \widehat{\pi}_3, x_4, \pi_4, \widetilde{\pi}_4) \\ \text{return} &\varrho \end{aligned}$	$\text{MDec}(\text{msk}, \varrho)$ <hr/> $\begin{aligned} \pi'_3 &:= \overline{I}_0.\text{Priv}(\overline{\text{sk}}_0, x_3); u := e_3 \cdot \pi'^{-1}_3 \\ \pi'_4 &:= \overline{I}_0.\text{Priv}(\overline{\text{sk}}_0, x_4) \\ \widehat{\pi}'_3 &:= I_3.\text{Priv}(\text{sk}_3, x_3, u); \widetilde{\pi}'_4 := I_4.\text{Priv}(\text{sk}_4, x_4, u) \\ \text{if } &(\widehat{\pi}'_3, \widetilde{\pi}'_4, \pi'_4) \neq (\widehat{\pi}_3, \widetilde{\pi}_4, \pi_4), \text{ return } \perp \\ \text{else} &\text{ return } u \end{aligned}$ $\text{Maul}(\varrho, r_\tau \in \overline{\Pi}_0)$ <hr/> $\begin{aligned} \widehat{\pi}'_3 &:= I_3.\text{RandH}(\widehat{\pi}_3, \widetilde{\pi}_4, 1_{\overline{\mathcal{R}}_x}, r_\tau) \\ \widetilde{\pi}'_4 &:= I_4.\text{RandH}(\widetilde{\pi}_4, \pi_4, 1_{\overline{\mathcal{R}}_x}, r_\tau) \\ \text{return} &\varrho' := (x_3, e_3 \cdot r_\tau, \widehat{\pi}'_3, x_4, \pi_4, \widetilde{\pi}'_4) \end{aligned}$ $\text{MRerand}(\varrho)$ <hr/> $\begin{aligned} r_3, r_4 &\leftarrow_{\$} \overline{\mathcal{R}}_x \\ x'_3 &:= \overline{I}_0.\text{RandX}(x_3, x_4, r_3); x'_4 := \overline{I}_0.\text{RandX}(x_4, x_4, r_4) \\ e'_3 &:= \overline{I}_0.\text{RandH}(e_3, \pi_4, r_3); \pi'_4 := \overline{I}_0.\text{RandH}(\pi_4, \pi_4, r_4) \\ \widehat{\pi}'_3 &:= I_3.\text{RandH}(\widehat{\pi}_3, \widetilde{\pi}_4, r_3, 1_{\overline{\Pi}_0}) \\ \widetilde{\pi}'_4 &:= I_4.\text{RandH}(\widetilde{\pi}_4, \pi_4, r_4, 1_{\overline{\Pi}_0}) \\ \text{return} &\varrho' := (x'_3, e'_3, \widehat{\pi}'_3, x'_4, \pi'_4, \widetilde{\pi}'_4) \end{aligned}$
--	---

Fig. 4. Generic rerandomizable and malleable encryption scheme MPKE

Table 2. Descriptions of Re-(T)-SPHF in the PKE. The first four rows describe the sets on which subset membership problems are defined, hash value spaces, tag spaces and hashing key spaces respectively. The rest of rows indicate certain algorithms in these Re-(T)-SPHF are required to be identical.

SPHF	I_0	I_1	I_2	\overline{I}_0	I_3	I_4
SMP	$(\mathcal{X}, \mathcal{L})$			$(\overline{\mathcal{X}}, \overline{\mathcal{L}})$		
Hash Value	Π_0		Π_1	$\overline{\Pi}_0$		Π_3
Tag	–	$\overline{\Pi}_0 \times \mathbb{Z}$		–	$\overline{\Pi}_0$	
Hashing Key	\mathcal{K}_0		\mathcal{K}_1	$\overline{\mathcal{K}}_0$		\mathcal{K}_3
Alg. ϕ	$I_0.\phi$		$I_1.\phi$	$\overline{I}_0.\phi$		$I_3.\phi$
Alg. RandX	$I_0.\text{RandX}$			$\overline{I}_0.\text{RandX}$		
Alg. RandT	–	$I_1.\text{RandT}$		–	$I_3.\text{RandT}$	

Descriptions of underlying SPHFs. We firstly describe the details of all the building blocks, i.e., the underlying Re-(T)-SPHFs, in Table 2.

For the Rand-RCCA security of the PKE, the underlying subset membership problems must be hard. Besides, we require that both I_0 and \overline{I}_0 are perfectly self-

rerandomizable and ST-Smooth₁; and I_1 is perfectly pairwise-rerandomizable on $\bar{I}_0 \times \{s\}$ for any $s \in \mathbb{Z}$, CPR-Smooth and PT-Smooth₁ with respect to I_2 ; and I_2 is perfectly self-rerandomizable on $\bar{I}_0 \times \{s\}$ for any $s \in \mathbb{Z}$ and CSR-Smooth; and I_3 is perfectly pairwise-rerandomizable, CPR-Smooth and PT-Smooth₁ with respect to I_4 ; and I_4 is perfectly self-rerandomizable and CSR-Smooth.

To ensure the consistency of rerandomization, we require that I_0 and \bar{I}_0 are linearly rerandomizable. Let ψ be an injection that maps \bar{I}_0 into \mathbb{Z} , $\mathcal{T}_1 = \bar{I}_0 \times \mathbb{Z}$ and $\mathcal{T}_3 = \bar{I}_0$. It is required that $I_1.\text{RandT}(\tau, r_\tau) = (r_\tau \cdot u, \psi(M))$ and $I_3.\text{RandT}(\tau', r_\tau) = r_\tau \cdot u$ for any $\tau = (u, \psi(M)) \in \mathcal{T}_1$, any $\tau' = u \in \mathcal{T}_3$ and any $r_\tau \in \bar{I}_0$. In algorithms Maul and MRerand, $1_{\bar{\mathcal{R}}_x}$ and $1_{\bar{I}_0}$ denote the identity elements in groups $\bar{\mathcal{R}}_x$ and \bar{I}_0 respectively.

Correctness. Below we analyze the correctness of the MPKE and then the PKE.

Theorem 1. *For any key pair (mpk, msk), any randomness $r_\tau \in \bar{I}_0$, any ciphertext ϱ and $\varrho' = \text{MRerand}(\text{Maul}(\varrho, r_\tau))$ in the scheme MPKE, we have*

$$\text{MDec}(\text{msk}, \varrho') = \begin{cases} r_\tau \cdot \text{MDec}(\text{msk}, \varrho), & \text{MDec}(\text{msk}, \varrho) \neq \perp \\ \perp, & \text{MDec}(\text{msk}, \varrho) = \perp \end{cases}.$$

Proof. Let $\varrho = (x_3, e_3, \hat{\pi}_3, x_4, \pi_4, \tilde{\pi}_4)$, $\text{msk} = (\bar{\text{sk}}_0, \text{sk}_3, \text{sk}_4)$ and $u = \text{MDec}(\text{msk}, \varrho)$. If $u \neq \perp$, then $e_3 \cdot u^{-1} = \bar{I}_0.\text{Priv}(\bar{\text{sk}}_0, x_3)$ holds and validity checking on ϱ passes. Let $\varrho' = (x'_3, e'_3, \hat{\pi}'_3, x'_4, \pi'_4, \tilde{\pi}'_4) = \text{MRerand}(\text{Maul}(\varrho, r_\tau))$. By the requirement on $I_3.\text{RandT}$, the linear rerandomizability of \bar{I}_0 and the consistency of rerandomization in \bar{I}_0 , I_3 and I_4 , let $u' = r_\tau \cdot u$, we have $e'_3 \cdot u'^{-1} = \bar{I}_0.\text{Priv}(\bar{\text{sk}}_0, x'_3)$ and the validity checking on ϱ' also passes. Thus, $\text{MDec}(\text{msk}, \varrho') = r_\tau \cdot u = r \cdot \text{MDec}(\text{msk}, \varrho)$.

If $u = \perp$, then $\pi_4 \neq \bar{I}_0.\text{Priv}(\bar{\text{sk}}_0, x_4)$, $\hat{\pi}_3 \neq I_3.\text{Priv}(\text{sk}_3, x_3, u)$ or $\tilde{\pi}_4 \neq I_4.\text{Priv}(\text{sk}_4, x_4, u)$ holds. In this case, the corresponding inequalities also hold in ciphertext ϱ' , then $\text{MDec}(\text{msk}, \varrho') = \perp$. ■

Theorem 2. *For any public/private key pair (PK, SK), any ciphertext ζ and $\zeta' = \text{Rerand}(\zeta)$ in the scheme PKE, we have $\text{Dec}(\text{SK}, \zeta) = \text{Dec}(\text{SK}, \zeta')$.*

Proof. Let $\zeta = (x_1, e_1, \hat{\pi}_1, x_2, \pi_2, \tilde{\pi}_2, \varrho)$ and $\zeta' = (x'_1, e'_1, \hat{\pi}'_1, x'_2, \pi'_2, \tilde{\pi}'_2, \varrho')$ be a rerandomized ciphertext of ζ . Let $\text{SK} = (\text{sk}_0, \text{sk}_1, \text{sk}_2, \text{msk})$, $u = \text{MDec}(\text{msk}, \varrho)$, $M = \text{Dec}(\text{SK}, \zeta)$ and $\tau = (u, \psi(M))$.

If $M \neq \perp$, then $u = \text{MDec}(\text{msk}, \varrho) \neq \perp$, $e_1 \cdot M^{-1} = I_0.\text{Priv}(\text{sk}_0, x_1)$ and the validity checking on ζ passes. By the requirement on $I_1.\text{RandT}$, the linear rerandomizability of I_0 and the consistency of rerandomization in I_0 , I_1 and I_2 , we have $e'_1 \cdot M^{-1} = I_0.\text{Priv}(\text{sk}_0, x'_1)$ and the validity checking on ϱ' passes. Thus, we have $\text{Dec}(\text{SK}, \zeta') = M$.

If $M = \perp$, then $u = \perp$, $\pi_2 \neq I_0.\text{Priv}(\text{sk}_0, x_2)$, $\hat{\pi}_1 \neq I_1.\text{Priv}(\text{sk}_1, x_1, \tau)$ or $\tilde{\pi}_2 \neq I_2.\text{Priv}(\text{sk}_2, x_2, \tau)$ holds. In this case, $u' = \perp$, by Theorem 1, or the corresponding inequalities hold in ζ' as well, and then $\text{Dec}(\text{SK}, \zeta') = \perp$. ■

5.2 Security Analysis

Noting that the scheme MPKE is a sub-scheme of PKE, below we will provide the security of PKE as the whole but will not separately give one regarding MPKE.

Theorem 3 (Perfect Rerandomization). *The scheme PKE is a perfectly rerandomizable encryption scheme.*

Proof. Given fixed plaintext M , key pair (PK, SK) , the distribution of the ciphertexts of M is determined by x_1, x_2, x_3, x_4 and u . Let ζ^* be a ciphertext in the support of $\text{Enc}(PK, M)$. Consider random variables $\zeta \leftarrow_{\$} \text{Enc}(PK, M)$ and $\zeta' \leftarrow_{\$} \text{Rerand}(\zeta^*)$. In ciphertext ζ , u is uniformly sampled from \overline{II}_0 , while $u' = r_\tau \cdot u^*$ in ζ' is also uniformly distributed on \overline{II}_0 as r_τ is randomly picked from \overline{II}_0 . By the perfect rerandomizability of \overline{I}_0, I_3 and I_4 , the distribution of ϱ and ϱ' is identical. Since I_0 is perfectly self-rerandomizable, the distribution of (x_1, e_1) (resp. (x_2, π_2)) in ζ is identical to that of (x'_1, e'_1) (resp. (x'_2, π'_2)) in ζ' . The distributions of $(x_1, \widehat{\pi}_1)$ and $(x'_1, \widehat{\pi}'_1)$ are identical by the perfect pairwise-rerandomizability of I_1 . Similarly, the distribution of $(x_2, \widetilde{\pi}_2)$ is the same as that of $(x'_2, \widetilde{\pi}'_2)$ by the perfect self-rerandomizability of I_2 . The 1-smoothness of all the Re-(T)-SPHF's guarantees that any (possibly unbounded) adversary is unable to generate a malicious ciphertext that is decryptable. Put it all together, the theorem follows. ■

Theorem 4 (RCCA Security). *For any $(\mathcal{X}, \mathcal{L})$ and $(\overline{\mathcal{X}}, \overline{\mathcal{L}})$ where subset membership problems are hard, the proposed PKE in Fig. 3 is RCCA-secure.*

Proof. We prove the RCCA security of the scheme PKE by constructing a sequence of games G_0 - G_3 and demonstrating the indistinguishability between them.

Game G_0 : This is the IND-RCCA game. Specifically, challenger generates key pair (PK, SK) via $KGen$, and sends PK to adversary \mathcal{A} . After querying decryption oracle \mathcal{DO}_{SK} , \mathcal{A} chooses two plaintexts M_0, M_1 . Then, challenger randomly picks $b \in \{0, 1\}$ and sends $\zeta^* \leftarrow_{\$} \text{Enc}(PK, M_b)$ to \mathcal{A} . Finally, \mathcal{A} outputs b' after querying guarded decryption oracle $\mathcal{GDO}_{SK}^{M_0, M_1}$.

Let S_i denote the event that $b = b'$ in game G_i , we have $\text{Adv}_{\mathcal{A}, \text{PKE}}^{\text{IND-RCCA}}(n) = |\Pr[S_0] - 1/2|$. Let the challenge ciphertext be $\zeta^* = (x_1^*, e_1^*, \widehat{\pi}_1^*, x_2^*, \pi_2^*, \widetilde{\pi}_2^*, \varrho^*)$ and $\varrho^* = (x_3^*, e_3^*, \widehat{\pi}_3^*, x_4^*, \pi_4^*, \widetilde{\pi}_4^*)$. Below we describe the modifications in G_1 - G_3 .

Game G_1 : This game is the same as G_0 except that challenge ciphertext ζ^* is generated by using secret key. Specifically, for the challenge ciphertext ζ^* , all the hash values are computed using hashing key. By the correctness of Re-(T)-SPHF's, same values would be computed in G_0 . The differences between G_0 and G_1 are only syntactical.

We call a ciphertext ζ bad if it is invalid (i.e., $\text{Dec}(SK, \zeta) = \perp$) or at least one of its elements is non-language (i.e., $x_1 \in \mathcal{X} \setminus \mathcal{L}, x_2 \in \mathcal{X} \setminus \mathcal{L}, x_3 \in \overline{\mathcal{X}} \setminus \overline{\mathcal{L}}$ or $x_4 \in \overline{\mathcal{X}} \setminus \overline{\mathcal{L}}$) unless it is a rerandomization of the challenge ciphertext. ■

Lemma 1. *In game G_1 , the decryption oracle rejects all the bad ciphertexts except with negligible probability.*

AltEnc(SK, $M \in \Pi_0$)	AltMEnc(msk, $u \in \bar{\Pi}_0$)
$x_1, x_2 \leftarrow_{\$} \mathcal{X} \setminus \mathcal{L}$	$x_3, x_4 \leftarrow_{\$} \bar{\mathcal{X}} \setminus \bar{\mathcal{L}}$
$u \leftarrow_{\$} \bar{\Pi}_0; \tau := (u, \psi(M))$	$e_3 := \bar{I}_0.\text{Priv}(\bar{\text{sk}}_0, x_3) \cdot u$
$e_1 := I_0.\text{Priv}(\text{sk}_0, x_1) \cdot M$	$\hat{\pi}_3 := I_3.\text{Priv}(\text{sk}_3, x_3, u)$
$\hat{\pi}_1 := I_1.\text{Priv}(\text{sk}_1, x_1, \tau)$	$\pi_4 := \bar{I}_0.\text{Priv}(\bar{\text{sk}}_0, x_4)$
$\pi_2 := I_0.\text{Priv}(\text{sk}_0, x_2)$	$\tilde{\pi}_4 := I_4.\text{Priv}(\text{sk}_4, x_4, u)$
$\tilde{\pi}_2 := I_2.\text{Priv}(\text{sk}_2, x_2, \tau)$	return $\varrho := (x_3, e_3, \hat{\pi}_3, x_4, \pi_4, \tilde{\pi}_4)$
$\varrho \leftarrow_{\$} \text{AltMEnc}(\text{msk}, u)$	
return $\zeta := (x_1, e_1, \hat{\pi}_1, x_2, \pi_2, \tilde{\pi}_2, \varrho)$	

Fig. 5. Modified encryption algorithms AltEnc and AltMEnc

Proof. First, querying a valid ciphertext ζ with $x_1, x_2 \in \mathcal{L}$ and $x_3, x_4 \in \bar{\mathcal{L}}$ does not reveal more information about the secret key SK.

Consider the first bad ciphertext ζ submitted to the decryption oracle. If at least one of its elements is non-language, by the 1-smoothness of $I_0, I_1, I_2, \bar{I}_0, I_3$ and I_4 , the corresponding hash value is uniformly distributed over appropriate domain and the probability that ζ is valid is negligible. If ζ is invalid, the decryption oracle rejects it with probability 1. Meanwhile, the rejection from decryption oracle rules out a negligible fraction of secret keys, and the correct secret key is still uniformly distributed among the rest of secret keys in adversary's view. Since the number of query is polynomial, the probability that adversary generates a "valid" bad ciphertext is negligible. ■

Game G_2 : This game is the same as G_1 except that challenge ciphertext ζ^* is generated with $x_3^*, x_4^* \leftarrow_{\$} \bar{\mathcal{X}} \setminus \bar{\mathcal{L}}$ and $x_1^*, x_2^* \leftarrow_{\$} \mathcal{X} \setminus \mathcal{L}$. That is, ζ^* is generated using AltEnc in Fig. 5. By the hardness of SMP on $(\bar{\mathcal{X}}, \bar{\mathcal{L}})$ and $(\mathcal{X}, \mathcal{L})$, games G_1 and G_2 are of computational indistinguishability. Here we omit the details of reduction.

Lemma 2. *In game G_2 , if the decryption oracles reject all the bad ciphertexts except with negligible probability, then the challenge ciphertext ζ^* is distributed independently of plaintext M_b and mask u^* , even given public key PK.*

Proof. Since $x_1^*, x_2^* \in \mathcal{X} \setminus \mathcal{L}$, by the pairwise-twin 1-smoothness of I_1 with respect to I_2 , $\hat{\pi}_1^*$ and $\tilde{\pi}_2^*$ are uniformly distributed over appropriate domains given $\text{pk}_1(\text{pk}_2)$. Similarly, $\hat{\pi}_3^*$ and $\tilde{\pi}_4^*$ are uniformly distributed over appropriate domains given $\text{pk}_3(\text{pk}_4)$ by the pairwise-twin 1-smoothness of I_3 with respect to I_4 . By the self-twin 1-smoothness of I_0 , both π_1^* and π_2^* are statistically close to random. Similarly, π_3^* and π_4^* are statistically close to random by the self-twin 1-smoothness of \bar{I}_0 . ■

By Lemma 1, in Phase 1, the decryption oracle rejects all the bad ciphertexts except with negligible probability. Thus, before Phase 2, u^* is uniformly distributed in adversary's view. This is crucial to the proof of Lemma 4.

Game G_3 : This game is the same as G_2 except that both decryption oracle \mathcal{DO}_{SK} (in Phase 1) and guarded decryption oracle $\mathcal{GDO}_{SK}^{M_0, M_1}$ (in Phase 2) return the output of alternate decryption algorithm AltDec (described below) that uses public keys and challenge ciphertext to decrypt ciphertexts instead of secret keys. We now prove that G_2 and G_3 are statistically indistinguishable. *Note that in this case AltDec is allowed to run in unbounded time.* In fact, this is essentially why AltDec is able to answer any decryption query using the public key and the challenge ciphertext only.

For any decryption query $\zeta = (x_1, e_1, \hat{\pi}_1, x_2, \pi_2, \tilde{\pi}_2, \varrho)$, we first describe the sub-algorithm AltMDec which is called by AltDec to decrypt $\varrho = (x_3, e_3, \hat{\pi}_3, x_4, \pi_4, \tilde{\pi}_4)$. Let $\varrho^* = (x_3^*, e_3^*, \hat{\pi}_3^*, x_4^*, \pi_4^*, \tilde{\pi}_4^*)$ denote the encryption of u^* in challenge ciphertext ζ^* . To decrypt ϱ , AltMDec performs as below.

- (i) Check that $x_3, x_4 \in \bar{\mathcal{L}}$. If not, go to (ii). Otherwise, let w_3, w_4 be the witnesses of x_3, x_4 , check that $\pi_4 = \bar{I}_0.\text{Pub}(\text{pk}_0, x_4, w_4)$ holds. If not, output \perp . Otherwise, compute $u = e_3 \cdot (\bar{I}_0.\text{Pub}(\text{pk}_0, x_3, w_3))^{-1}$, and check that $\hat{\pi}_3 = I_3.\text{Pub}(\text{pk}_3, x_3, w_3, u)$ and $\tilde{\pi}_4 = I_4.\text{Pub}(\text{pk}_4, x_4, w_4, u)$ hold. If not, output \perp . Otherwise, output $(\sigma = u, s = 0)$.
- (ii) If AltMDec is called in Phase 1, output \perp . Otherwise, check that there exist $r_3, r_4 \in \bar{\mathcal{R}}_x$ and $r_\tau \in \bar{I}_0$ such that $\varrho = \text{MRerand}(\text{Maul}(\varrho^*, r_\tau))$. If r_3, r_4 or r_τ does not exist, output \perp . Otherwise, output $(\sigma = r_\tau, s = 1)$.

The correctness of AltMDec is proved in Lemma 3.

Lemma 3. *Let (mpk, msk) be a public/secret key pair of the MPKE and ϱ^* be a ciphertext generated using AltMEnc . Let $(\sigma, s) = \text{AltMDec}(\text{mpk}, \varrho^*, \varrho)$, if $(\sigma, s) \neq \perp$, then $\text{MDec}(\text{msk}, \varrho) = \sigma \cdot \text{MDec}(\text{msk}, \varrho^*)^s$.*

Proof. If $s = 0$, ϱ is a fresh encryption of u with $x_3, x_4 \in \bar{\mathcal{L}}$. By the correctness of \bar{I}_0, I_3 and I_4 , MDec also decrypts ϱ into u . If $s = 1$, ϱ is a derivative ciphertext of ϱ^* . Although ϱ and ϱ^* both are not generated by MEnc , one can verify that $\text{MDec}(\text{msk}, \varrho) = r_\tau \cdot u^* = r_\tau \cdot \text{MDec}(\text{msk}, \varrho^*)$. ■

Now we are ready to describe AltDec . Let $\zeta^* = (x_1^*, e_1^*, \hat{\pi}_1^*, x_2^*, \pi_2^*, \tilde{\pi}_2^*, \varrho^*)$ be the challenge ciphertext. AltDec then decrypts $\zeta = (x_1, e_1, \hat{\pi}_1, x_2, \pi_2, \tilde{\pi}_2, \varrho)$ with PK and ζ^* as below.

- (i) Compute $(\sigma, s) = \text{AltMDec}(\text{mpk}, \varrho^*, \varrho)$. If AltMDec returns \perp , then also return \perp .
- (ii) If $s = 0$, then $\sigma = u$. Check that there exist message M and witnesses w_1, w_2 such that $x_1, x_2 \in \mathcal{L}$ and

$$\begin{aligned} e_1 &= I_0.\text{Pub}(\text{pk}_0, x_1, w_1) \cdot M & \pi_2 &= I_0.\text{Pub}(\text{pk}_0, x_2, w_2) \\ \hat{\pi}_1 &= I_1.\text{Pub}(\text{pk}_1, x_1, w_1, \tau) & \tilde{\pi}_2 &= I_2.\text{Pub}(\text{pk}_2, x_2, w_2, \tau), \end{aligned}$$

where $\tau = (u, \psi(M))$. If not, output \perp . If $M \notin \{M_0, M_1\}$, output M ; otherwise, output **replay**.

(iii) If $s = 1$, then $\sigma = r_\tau$. Check that there exist randomness $r_1, r_2 \in \mathcal{R}_x$ such that following equalities hold.

$$\begin{aligned} x_1 &= I_0.\text{RandX}(x_1^*, x_2^*, r_1) & x_2 &= I_0.\text{RandX}(x_2^*, x_2^*, r_2) \\ e_1 &= I_0.\text{RandH}(e_1^*, \pi_2^*, r_1) & \pi_2 &= I_0.\text{RandH}(\pi_2^*, \pi_2^*, r_2) \\ \hat{\pi}_1 &= I_1.\text{RandH}(\hat{\pi}_1^*, \tilde{\pi}_2^*, r_1, r_\tau) & \tilde{\pi}_2 &= I_2.\text{RandH}(\tilde{\pi}_2^*, \tilde{\pi}_2^*, r_2, r_\tau). \end{aligned}$$

If not, output \perp . Otherwise, output **replay**.

Lemma 4. *The output of \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathcal{G}_3 agrees with the output of \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathcal{G}_2 with overwhelming probability.*

Proof. In the cases where \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathcal{G}_3 outputs M , \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathcal{G}_2 also outputs M by Lemma 3 and the correctness of decryption. Similarly, when $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$ in \mathcal{G}_3 outputs **replay**, $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$ in \mathcal{G}_2 also outputs **replay** by Lemma 3 and correctness of decryption and rerandomization.

We now prove that when \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathcal{G}_3 outputs \perp on query ζ , \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathcal{G}_2 also would output \perp with overwhelming probability. That is, when **AltDec** outputs \perp , **Dec** also would output \perp with overwhelming probability. Let $\zeta^* = (x_1^*, e_1^*, \hat{\pi}_1^*, x_2^*, \pi_2^*, \tilde{\pi}_2^*, \varrho^*)$ denote the challenge ciphertext where $\varrho^* = (x_3^*, e_3^*, \hat{\pi}_3^*, x_4^*, \pi_4^*, \tilde{\pi}_4^*)$ and $\zeta = (x_1, e_1, \hat{\pi}_1, x_2, \pi_2, \tilde{\pi}_2, \varrho)$ denote the decryption query input where $\varrho = (x_3, e_3, \hat{\pi}_3, x_4, \pi_4, \tilde{\pi}_4)$.

Case 1. If **AltDec** outputs \perp due to **AltMDec** returning \perp , there are following possible sub-cases.

- In Phase 1, $x_3 \notin \bar{\mathcal{L}}$ or $x_4 \notin \bar{\mathcal{L}}$. By the 1-smoothness of \bar{I}_0 , $\pi_3 = e_3 \cdot u^{-1}$ or π_4 is statistically close to random, and thus ζ will be rejected by **Dec** with overwhelming probability.
- In Phase 2, $r_3, r_4 \in \bar{\mathcal{R}}_x$ or $r_\tau \in \bar{\Pi}_0$ does not exist for $\varrho = \text{MRerand}(\text{Maul}(\varrho^*, r_\tau))$ with x_3 or $x_4 \notin \bar{\mathcal{L}}$. If r_τ does not exist, by the CPR-Smooth of I_3 or CSR-Smooth of I_4 , $\hat{\pi}_3$ or $\tilde{\pi}_4$ is close to random, as x_3 or $x_4 \notin \bar{\mathcal{L}}$. If r_3 does not exist and $x_3 \notin \bar{\mathcal{L}}$, $\hat{\pi}_3$ is close to random by the CPR-Smooth of I_3 . If r_4 does not exist and $x_4 \notin \bar{\mathcal{L}}$, $\tilde{\pi}_4$ is close to random by the CSR-Smooth of I_4 . If r_3 does not exist and $x_3 \in \bar{\mathcal{L}}$, then $x_4 \notin \bar{\mathcal{L}}$. In this case, we assume that there exists r_4 such that $x_4 = \bar{I}_0.\text{RandX}(x_4^*, x_4^*, r_4)$. Since u^* is uniformly sampled from $\bar{\Pi}_0$ at random, the underlying u of $\tilde{\pi}_4$ equals to $r_\tau \cdot u^*$ which is uniformly distributed over $\bar{\Pi}_0$. Then, $\hat{\pi}_3$ is close to random, as u is uniformly distributed and $\hat{\pi}_3$ is independent of $\hat{\pi}_3^*$. Similarly, we can prove that $\tilde{\pi}_4$ is close to random when r_4 does not exist, r_3 exists, $x_4 \in \bar{\mathcal{L}}$ and $x_3 \notin \bar{\mathcal{L}}$.
- In both Phase 1 and 2, $\pi_4 \neq \bar{I}_0.\text{Pub}(\bar{\text{pk}}_0, x_4, w_4)$, $\hat{\pi}_3 \neq I_3.\text{Pub}(\text{pk}_3, x_3, w_3, u)$ or $\tilde{\pi}_4 \neq I_4.\text{Pub}(\text{pk}_4, x_4, w_4, u)$ holds. Obviously, **MDec** would reject ϱ and **Dec** would reject ζ .

Case 2. Suppose that $(\sigma, s) = \text{AltMDec}(\text{mpk}, \varrho^*, \varrho)$ and $(\sigma, s) \neq \perp$. There are following sub-cases where **AltDec** outputs \perp .

- In Phase 1, $(\sigma, s) = (u, 0)$ and x_1 or $x_2 \notin \mathcal{L}$. By the 1-smoothness of I_0, I_1 and $I_2, \pi_2, \hat{\pi}_1$ or $\tilde{\pi}_2$ is statistically close to random. Suppose $x_1, x_2 \in \mathcal{L}$ and $\text{pk}_0, \text{pk}_1(\text{pk}_2)$ are fixed. If any equation in decryption rule (ii) of AltDec does not hold for any $M \in \Pi_0, \zeta$ would be rejected due to the validity checking.
- In Phase 2, $(\sigma, s) = (u, 0)$ and x_1 or $x_2 \notin \mathcal{L}$. If $x_1 = I_0.\text{RandX}(x_1^*, x_2^*, r_1)$ or $x_2 = I_0.\text{RandX}(x_2^*, x_2^*, r_2)$, the underlying tag $\tau = (u, \psi(M))$ of $\hat{\pi}_1$ or $\tilde{\pi}_2$ which is derived from $\hat{\pi}_1^*$ and $\tilde{\pi}_2^*$ via $I_1.\text{RandH}$ or $I_2.\text{RandH}$ would be related to $\tau^* = (u^*, \psi(M^*))$ where u^* is uniformly distributed over $\bar{\Pi}_0$. However, $s = 0$ indicates that the value of u is fixed and $u = \sigma$. Thus, the validity checking on ζ would fail. Otherwise, $x_1 \neq I_0.\text{RandX}(x_1^*, x_2^*, r_1)$ and $x_2 \neq I_0.\text{RandX}(x_2^*, x_2^*, r_2)$. Given fixed $\text{pk}_1, \hat{\pi}_1^*$ and $\tilde{\pi}_2^*$, the value of $\hat{\pi}_1$ is statistically close to random as I_1 is CPR-Smooth.
- In Phase 1 and 2, $(\sigma, s) = (u, 0)$ and $x_1, x_2 \in \mathcal{L}$. If equations in rule (ii) of AltDec do not hold simultaneously for any $M \in \Pi_0$, the validity checking on ζ in Dec would fail.
- In Phase 2, $(\sigma, s) = (r_\tau, 1)$, and there do not exist $r_1, r_2 \in \mathcal{R}_x$ such that equations in decryption rule (iii) of AltDec hold at the same time. If $x_1 \neq I_0.\text{RandX}(x_1^*, x_2^*, r_1)$ for any $r_1 \in \mathcal{R}_x$ or $\tau \neq I_0.\text{RandT}(\tau^*, r_\tau)$, due to the fact that I_1 is CPR-Smooth, $\hat{\pi}_1$ is statistically indistinguishable from random hash value given fixed $\text{pk}_1, \hat{\pi}_1^*$ and $\tilde{\pi}_2^*$. Similarly, if $x_2 \neq I_0.\text{RandX}(x_2^*, x_2^*, r_2)$ for any $r_2 \in \mathcal{R}_x$ or $\tau \neq I_0.\text{RandT}(\tau^*, r_\tau)$, due to the fact that I_2 is CSR-Smooth, $\tilde{\pi}_2$ is statistically close to random hash value given fixed pk_2 and $\tilde{\pi}_2^*$. Suppose that $x_1 = I_0.\text{RandX}(x_1^*, x_2^*, r_1), x_2 = I_0.\text{RandX}(x_1^*, x_2^*, r_2)$ and $\tau = I_0.\text{RandT}(\tau^*, r_\tau)$. If equations in rule (iii) of AltDec do not hold simultaneously, the validity checking on ζ in Dec would fail.

In conclusion, The output of \mathcal{DO}_{SK} (resp. $\mathcal{GDO}_{\text{SK}}^{M_0, M_1}$) in \mathbf{G}_3 is the same as that in \mathbf{G}_2 in every case with overwhelming probability. ■

Lemma 5. $\Pr[S_3] = 1/2$.

Proof. Note that AltMDec and AltDec do not use secret key to perform decryption. The decryption oracle responses in game \mathbf{G}_3 do not provide extra information about secret key besides public key and challenge ciphertext ζ^* generated using AltEnc. Lemma 2 shows that ζ^* is distributed independently of bit b , from which the lemma follows. ■

Putting it all together, the theorem follows. ■

Theorem 5 (RCCA Receiver-Anonymity). *For any $(\mathcal{X}, \mathcal{L})$ and $(\bar{\mathcal{X}}, \bar{\mathcal{L}})$ where subset membership problems are hard, the proposed PKE in Fig. 3 is RCCA receiver-anonymous.*

Proof. We prove the receiver-anonymity of PKE by constructing a sequence of games \mathbf{G}_0 - \mathbf{G}_3 and demonstrating the indistinguishability between them.

Game \mathbf{G}_0 : This is the ANON-RCCA game. Specifically, challenger generates two key pairs $(\text{PK}_0, \text{SK}_0)$ and $(\text{PK}_1, \text{SK}_1)$ via KGen, and sends $(\text{PK}_0, \text{PK}_1)$ to

adversary \mathcal{A} . After querying decryption oracle $\mathcal{DO}_{\text{SK}_0, \text{SK}_1}$, \mathcal{A} chooses a plaintext M . Then, challenger randomly picks $b \in \{0, 1\}$ and sends $\zeta^* \leftarrow_{\mathfrak{s}} \text{Enc}(\text{PK}_b, M)$ to \mathcal{A} . Finally, \mathcal{A} outputs b' after querying guarded decryption oracle $\mathcal{GDO}_{\text{SK}_0, \text{SK}_1}^M$.

Let S_i denote the event that $b = b'$ in game G_i , we have $\text{Adv}_{\mathcal{A}, \text{PKE}}^{\text{ANON-RCCA}}(n) = |\Pr[S_0] - 1/2|$.

Game G_1 : This game is the same as G_0 except that challenge ciphertext ζ^* is generated by using secret key SK_b . According to the analysis in Theorem 4, game G_1 is identical to G_0 by the correctness of SPHF's.

Game G_2 : This game is the same as G_1 except that challenge ciphertext ζ^* is generated with $x_3^*, x_4^* \leftarrow_{\mathfrak{s}} \overline{\mathcal{X}} \setminus \overline{\mathcal{L}}$ and $x_1^*, x_2^* \leftarrow_{\mathfrak{s}} \mathcal{X} \setminus \mathcal{L}$. That is, ζ^* is generated using AltEnc in Fig. 5. By the hardness of SMP on $(\overline{\mathcal{X}}, \overline{\mathcal{L}})$ and $(\mathcal{X}, \mathcal{L})$, games G_1 and G_2 are of computational indistinguishability.

Game G_3 : This game is the same as G_2 except that both decryption oracle $\mathcal{DO}_{\text{SK}_0, \text{SK}_1}$ (in Phase 1) and guarded decryption oracle $\mathcal{GDO}_{\text{SK}_0, \text{SK}_1}^M$ (in Phase 2) work as follows. First, it runs alternative decryption algorithm AltDec^* , which is the same as AltDec in Theorem 4 except that it outputs `replay` when decryption result equals to M , with PK_0 and PK_1 respectively. If AltDec^* outputs `replay`, it returns `replay`, otherwise, it returns the results of running AltDec^* . By Lemma 4, the output of $\mathcal{DO}_{\text{SK}_0, \text{SK}_1}(\mathcal{GDO}_{\text{SK}_0, \text{SK}_1}^M)$ in G_3 agrees with the output of $\mathcal{DO}_{\text{SK}_0, \text{SK}_1}(\mathcal{GDO}_{\text{SK}_0, \text{SK}_1}^M)$ in G_2 with overwhelming probability. Thus, games G_2 and G_3 are statistically indistinguishable.

Note that AltDec^* does not use secret key to perform decryption. The decryption oracle responses in game G_3 do not provide extra information about secret key SK_b besides public keys PK_0, PK_1 and challenge ciphertext ζ^* generated using AltEnc . By Lemma 2, ζ^* is distributed independently of PK_b . Thus, we have $\Pr[S_3] = 1/2$, from which the theorem follows. \blacksquare

6 Instantiations

In this section, we show how to instantiate our framework from the k -LIN assumption. More generally, it could be constructed from graded rings [3] and we provide the details in the full version [29].

6.1 Regular SPHF from k -LIN Assumption

Let \mathbb{G} be a cyclic group with prime order p . The k -LIN assumption says that $[\mathbf{r}^\top \mathbf{g}_{k+1}]$ is pseudorandom given $[\mathbf{g}^\top]$, $[g_{k+1}]$, $[\mathbf{r}^\top \mathbf{G}]$ where $\mathbf{r}, \mathbf{g} \leftarrow_{\mathfrak{s}} \mathbb{Z}_p^k$, $g_{k+1} \leftarrow_{\mathfrak{s}} \mathbb{Z}_p$ and $\mathbf{G} = \text{diag}(\mathbf{g}^\top) \in \mathbb{Z}_p^{k \times k}$, $\mathbf{g}_{k+1} = (g_{k+1}, \dots, g_{k+1})^\top \in \mathbb{Z}_p^k$.

Let element set $\mathcal{X} = \{[\mathbf{x}^\top] | \mathbf{x} \in \mathbb{Z}_p^{k+1}\}$ and $\mathcal{L} = \{[\mathbf{w}^\top \mathbf{P}] | \mathbf{w} \in \mathbb{Z}_p^k\}$ where $\mathbf{P} = (\mathbf{G} \ \mathbf{g}_{k+1}) \in \mathbb{Z}_p^{k \times (k+1)}$. Below is a regular SPHF from k -LIN assumption.

- $\text{Setup}(1^n)$. Let $\mathcal{K} = \mathbb{Z}_p^{k+1}$, $\mathcal{H} = \mathbb{G}$ and $\mathcal{T} = \emptyset$. Since the tag space is empty, $H_{(\cdot)} : \mathcal{X} \rightarrow \mathbb{G}$ is an efficient hash function family indexed by $\text{sk} \in \mathbb{Z}_p^{k+1}$.
- $\phi(\text{sk})$. For $\text{sk} = \mathbf{a} \in \mathbb{Z}_p^{k+1}$, outputs $\text{pk} = [\mathbf{P}\mathbf{a}] \in \mathbb{G}^k$.

- $\text{Priv}(\text{sk}, x)$. For $\text{sk} = \mathbf{a} \in \mathbb{Z}_p^{k+1}$ and $x = [\mathbf{x}^\top] \in \mathcal{X}$, outputs $\pi = [\mathbf{x}^\top \mathbf{a}] \in \mathbb{G}$.
- $\text{Pub}(\text{pk}, x, w)$. For $\text{pk} = [\mathbf{Pa}] \in \mathbb{G}^k$ and $x = [\mathbf{w}^\top \mathbf{P}] \in \mathcal{L}$ with witness $\mathbf{w} \in \mathbb{Z}_p^k$, outputs $\pi = [\mathbf{w}^\top (\mathbf{Pa})] \in \mathbb{G}$.

Since $[\mathbf{w}^\top (\mathbf{Pa})] = [(\mathbf{w}^\top \mathbf{P})\mathbf{a}]$, the correctness of SPHF holds. For any $x \notin \mathcal{L}$ and $\text{pk} = [\mathbf{Pa}]$, vector \mathbf{x}^\top is not in the linear span of \mathbf{P} , then hash value $H_{\text{sk}}(x) = [\mathbf{x}^\top \mathbf{a}]$ is independent from $\text{pk} = [\mathbf{Pa}]$. This guarantees the 1-smoothness.

6.2 Instantiating the Underlying Re-(T)-SPHFs of Our Framework

(1) Construction of I_0 and \bar{I}_0 . The algorithms $(I_0.\text{Setup}, I_0.\phi, I_0.\text{Priv}, I_0.\text{Pub})$ are the same as those of regular SPHF from k -LIN assumption, and thus the 1-smoothness of I_0 is obvious. Below we provide the remaining algorithms, i.e., $I_0.\text{RandX}$ and $I_0.\text{RandH}$.

- $I_0.\text{RandX}(x, x', r_x)$. For $x = [\mathbf{x}^\top], x' = [\mathbf{x}'^\top] \in \mathcal{X}$ and $r_x \in \mathbb{Z}_p$, outputs $x^* = [\mathbf{x}^\top + r_x \mathbf{x}'^\top]$.
- $I_0.\text{RandH}(\pi, \pi', r_x)$. For $\pi = [\mathbf{x}^\top \mathbf{a}], \pi' = [\mathbf{x}'^\top \mathbf{a}] \in \mathbb{G}$ and $r_x \in \mathbb{Z}_p$, outputs $\pi^* = \pi \cdot (\pi')^{r_x} = [\mathbf{x}^\top \mathbf{a} + r_x \mathbf{x}'^\top \mathbf{a}]$.

Since $\pi^* = [(\mathbf{x}^\top + r_x \mathbf{x}'^\top)\mathbf{a}] = I_0.\text{Priv}(\text{sk}, I_0.\text{RandX}(x, x', r_x))$, the correctness of rerandomization holds. For any $\pi, \pi', \Delta \in \mathbb{G}$ and any $r_x \in \mathbb{Z}_p$, we have $I_0.\text{RandH}(\pi \cdot \Delta, \pi', r_x) = (\pi \cdot \Delta) \cdot (\pi')^{r_x} = (\pi \cdot (\pi')^{r_x}) \cdot \Delta = I_0.\text{RandH}(\pi, \pi', r_x) \cdot \Delta$ and I_0 is linearly rerandomizable. Due to lack of space, the proofs of following theorems appear in the full version [29].

Theorem 6. I_0 is perfectly self-rerandomizable.

Theorem 7. I_0 is ST-Smooth₁ when $k \geq 2$.

The construction of \bar{I}_0 is exactly the same as I_0 . In concrete scheme, it is associated with $\bar{\mathcal{X}}$ and NP-language $\bar{\mathcal{L}}$ that are defined over $\bar{\mathbb{G}}^{k+1}$ where $\bar{\mathbb{G}}$ is a cyclic group with prime order q and a subgroup of \mathbb{Z}_p^* . Specifically, $\bar{\mathcal{X}} = \{[\bar{\mathbf{x}}^\top] | \bar{\mathbf{x}} \in \mathbb{Z}_q^{k+1}\}$, and $\bar{\mathcal{L}} = \{[\bar{\mathbf{w}}^\top \bar{\mathbf{P}}] | \bar{\mathbf{w}} \in \mathbb{Z}_q^k\}$ where $\bar{\mathbf{P}} = (\bar{\mathbf{G}} \bar{\mathbf{g}}_{k+1}) \in \mathbb{Z}_q^{k \times (k+1)}$, $\bar{\mathbf{G}} = \text{diag}(\bar{\mathbf{g}}^\top) \in \mathbb{Z}_q^{k \times k}$, $\bar{\mathbf{g}}_{k+1} = (\bar{g}_{k+1}, \dots, \bar{g}_{k+1})^\top \in \mathbb{Z}_q^k$, $\bar{\mathbf{g}} \leftarrow_s \mathbb{Z}_q^k, \bar{g}_{k+1} \leftarrow_s \mathbb{Z}_q$.

(2) Construction of I_1 and I_2 . We first describe the framework of I_1 as below.

- $I_1.\text{Setup}(1^n)$. Let $\mathcal{K}_1 = (\mathbb{Z}_p^{k+1})^4$, $\Pi_1 = \mathbb{G}^2$, $\mathcal{T}_1 = \bar{\mathbb{G}} \times \mathbb{Z}_p^*$. Pick $\lambda_1, \lambda_2 \leftarrow_s \mathbb{Z}_p^k$ with $\lambda_1 \neq \lambda_2$, $\text{ax} = (\lambda_1, \lambda_2)$. $\hat{H}(\cdot) : \mathcal{X} \times \mathcal{T}_1 \rightarrow \mathbb{G}^2$ is indexed by $\text{sk}_1 \in \mathcal{K}_1$ and ax .
- $I_1.\phi(\text{sk}_1)$. For $\text{sk}_1 = (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}) \in (\mathbb{Z}_p^{k+1})^4$, outputs

$$\text{pk}_1 = ([\mathbf{Pb}], [\mathbf{Pc}], [\mathbf{Pd}], [\mathbf{Pe}]).$$

- $I_1.\text{Priv}(\text{sk}_1, x, \tau)$. For $\text{sk}_1 = (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e})$, $x = [\mathbf{x}^\top]$ and $\tau = (\tau_0, \tau_1)$, outputs hash value $\pi = \hat{H}_{\text{sk}_1}(x, \tau) = (\pi_1, \pi_2) =$

$$([\mathbf{x}^\top + \lambda_1^\top \mathbf{P}](\tau_0(\mathbf{b} + \tau_1 \mathbf{c})), [\mathbf{x}^\top + \lambda_2^\top \mathbf{P}](\tau_0(\mathbf{d} + \tau_1 \mathbf{e}))].$$

- $I_1.\text{Pub}(\mathbf{pk}_1, x, w, \tau)$. For $\mathbf{pk}_1 = ([\mathbf{Pb}], [\mathbf{Pc}], [\mathbf{Pd}], [\mathbf{Pe}])$, $x = [\mathbf{w}^\top \mathbf{P}]$ with witness \mathbf{w} and $\tau = (\tau_0, \tau_1)$, outputs $\pi = \widehat{H}_{\mathbf{sk}_1}(x, \tau) = (\pi_1, \pi_2) =$

$$([\mathbf{w}^\top + \boldsymbol{\lambda}_1^\top](\tau_0(\mathbf{Pb} + \tau_1 \mathbf{Pc}))], [(\mathbf{w}^\top + \boldsymbol{\lambda}_2^\top)(\tau_0(\mathbf{Pd} + \tau_1 \mathbf{Pe}))]).$$

- $I_1.\text{RandX}(x, x', r_x)$. For $x = [\mathbf{x}^\top]$, $x' = [\mathbf{x}'^\top]$ and $r_x \in \mathbb{Z}_p$, outputs $x^* = [\mathbf{x}^\top + r_x \mathbf{x}'^\top]$.
- $I_1.\text{RandT}(\tau, r_\tau)$. For $\tau = (\tau_0, \tau_1)$ and $r_\tau \in \mathbb{Z}_p$, outputs $\tau^* = (r_\tau \cdot \tau_0, \tau_1)$.
- $I_1.\text{RandH}(\pi, \pi', r_x, r_\tau)$. For $\pi = (\pi_1, \pi_2)$, $\pi' = (\pi'_1, \pi'_2)$, $r_x \in \mathbb{Z}_p$ and $r_\tau \in \mathbb{Z}_p$, outputs $\pi^* = ((\pi_1 \cdot (\pi'_1)^{r_x})^{r_\tau}, (\pi_2 \cdot (\pi'_2)^{r_x})^{r_\tau})$.

As for I_2 , its algorithms $I_2.\phi$, $I_2.\text{RandX}$, $I_2.\text{RandT}$ and $I_2.\text{RandH}$ are the same as $I_1.\phi$, $I_1.\text{RandX}$, $I_1.\text{RandT}$ and $I_1.\text{RandH}$. Besides, $I_2.\text{Setup}$ is the same as $I_1.\text{Setup}$ except that \mathbf{ax} is null and the hash function family is $\widetilde{H}_{(\cdot)} : \mathcal{X} \times \mathcal{T}_2 \rightarrow \mathbb{G}^2$ where $\mathcal{T}_2 = \mathcal{T}_1$. $I_2.\text{Priv}$ and $I_2.\text{Pub}$ are equivalent to $I_1.\text{Priv}$ and $I_1.\text{Pub}$ with $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_2 = \mathbf{0}$.

- $I_2.\text{Priv}(\mathbf{sk}_2, x, \tau)$. For $\mathbf{sk}_2 = (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}) \in (\mathbb{Z}_p^{k+1})^4$, $x = [\mathbf{x}^\top]$ and $\tau = (\tau_0, \tau_1)$, outputs hash value $\pi = \widetilde{H}_{\mathbf{sk}_2}(x, \tau) = (\pi_1, \pi_2) =$

$$([\mathbf{x}^\top(\tau_0(\mathbf{b} + \tau_1 \mathbf{c}))], [\mathbf{x}^\top(\tau_0(\mathbf{d} + \tau_1 \mathbf{e}))]).$$

- $I_2.\text{Pub}(\mathbf{pk}_2, x, w, \tau)$. For $\mathbf{pk}_2 = ([\mathbf{Pb}], [\mathbf{Pc}], [\mathbf{Pd}], [\mathbf{Pe}])$, $x = [\mathbf{w}^\top \mathbf{P}]$ with witness \mathbf{w} and $\tau = (\tau_0, \tau_1)$, outputs $\pi = \widetilde{H}_{\mathbf{sk}_2}(x, \tau) = (\pi_1, \pi_2) =$

$$([\mathbf{w}^\top(\tau_0(\mathbf{Pb} + \tau_1 \mathbf{Pc}))], [\mathbf{w}^\top(\tau_0(\mathbf{Pd} + \tau_1 \mathbf{Pe}))]).$$

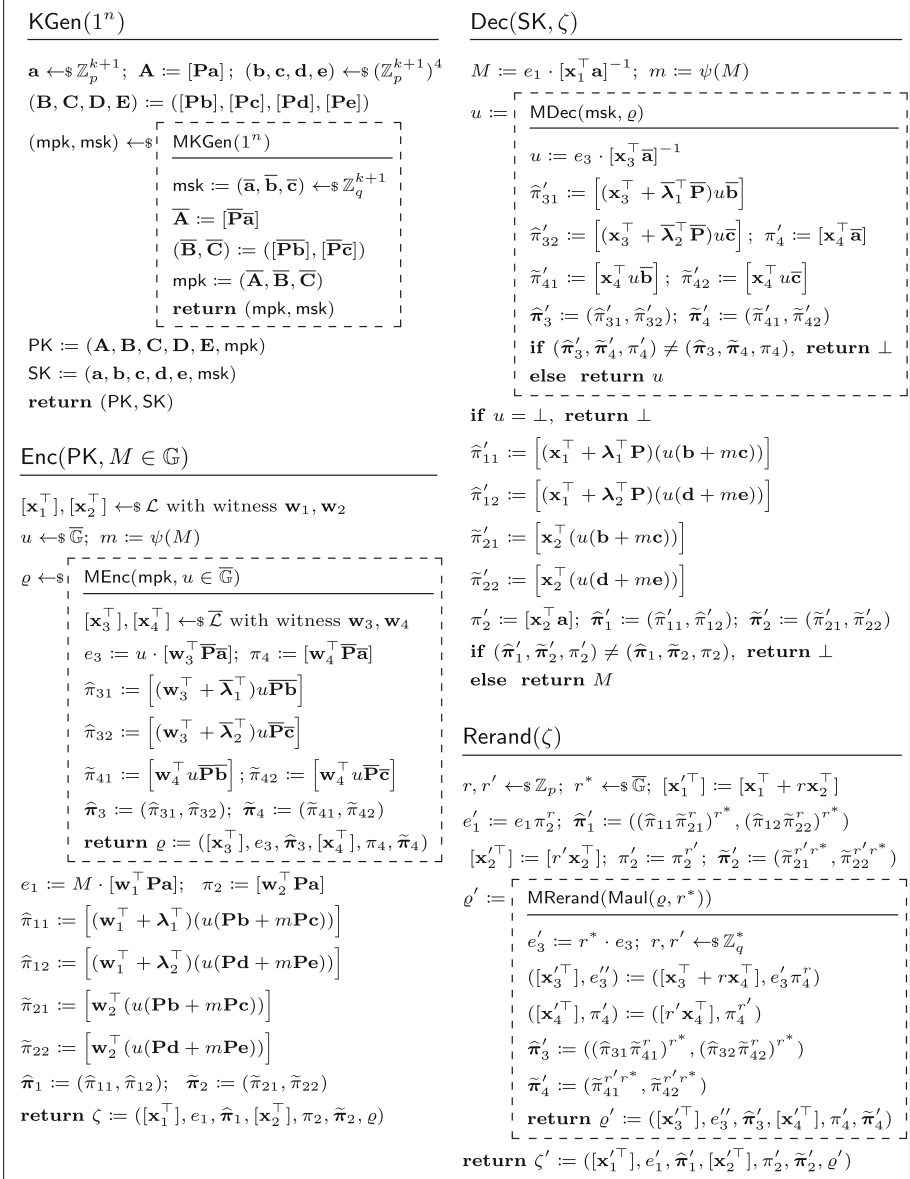
One can verify the correctness of I_1 and I_2 easily. For any $x \notin \mathcal{L}$, any $\tau \in \mathcal{T}_1$ and $\mathbf{pk}_1 = ([\mathbf{Pb}], [\mathbf{Pc}], [\mathbf{Pd}], [\mathbf{Pe}])$, vector \mathbf{x}^\top is not in the linear span of \mathbf{P} , then $(([\mathbf{x}^\top + \boldsymbol{\lambda}_1^\top \mathbf{P}](\tau_0(\mathbf{b} + \tau_1 \mathbf{c}))], [(\mathbf{x}^\top + \boldsymbol{\lambda}_2^\top \mathbf{P})(\tau_0(\mathbf{d} + \tau_1 \mathbf{e}))])$ is independent of \mathbf{pk}_1 , from which the 1-smoothness property holds for both I_1 and I_2 . As for the correctness of rerandomization, we consider $\pi = \widehat{H}_{\mathbf{sk}_1}(x, \tau)$ and $\pi' = \widetilde{H}_{\mathbf{sk}_2}(x', \tau)$ as I_1 is rerandomizable with respect to I_2 . For $r_x, r_\tau \in \mathbb{Z}_p$, one can verify that rerandomized hash value $\pi^* = I_1.\text{RandH}(\pi, \pi', r_x, r_\tau) = I_1.\text{Priv}(\mathbf{sk}_1, x^*, \tau^*)$ where $x^* = I_1.\text{RandX}(x, x', r_x)$ and $\tau^* = I_1.\text{RandT}(\tau, r_\tau)$. This also holds for $\pi = \widetilde{H}_{\mathbf{sk}_2}(x, \tau)$ and $\pi' = \widetilde{H}_{\mathbf{sk}_2}(x', \tau)$. The proofs of following theorems are provided in the full version [29].

Theorem 8. *Let $\mathcal{T}_1(s) = \overline{\mathbb{G}} \times \{s\} \subseteq \mathcal{T}_1$ with $s \in \mathbb{Z}_p^*$. I_1 is perfectly pairwise-rerandomizable on $\mathcal{T}_1(s)$ with respect to I_2 for any $s \in \mathbb{Z}_p^*$.*

Theorem 9. *Let $\mathcal{T}_2(s) = \overline{\mathbb{G}} \times \{s\} \subseteq \mathcal{T}_2$ with $s \in \mathbb{Z}_p^*$. I_2 is perfectly self-rerandomizable on $\mathcal{T}_2(s)$ for any $s \in \mathbb{Z}_p^*$.*

Theorem 10. *I_1 is PT-Smooth₁ with respect to I_2 when $k \geq 2$.*

Theorem 11. *I_1 is CPR-Smooth with respect to I_2 .*

Fig. 6. k -LIN-based anonymous Rand-RCCA-secure scheme PKE

Theorem 12. I_2 is CSR-Smooth.

(3) Construction of I_3 and I_4 . We first describe the framework of I_3 as below.

- $I_3.\text{Setup}(1^n)$. Let $\mathcal{K}_3 = (\mathbb{Z}_q^{k+1})^2$, $\Pi_3 = \overline{\mathbb{G}}^2$ and $\mathcal{T}_3 = \overline{\mathbb{G}}$. Pick $\overline{\lambda}_1, \overline{\lambda}_2 \leftarrow_{\$} \mathbb{Z}_q^k$ with $\overline{\lambda}_1 \neq \overline{\lambda}_2$, $\text{ax} = (\overline{\lambda}_1, \overline{\lambda}_2)$ and $\widehat{H}_{(\cdot)} : \overline{\mathcal{X}} \times \mathcal{T}_3 \rightarrow \overline{\mathbb{G}}^2$ is indexed by $\text{sk}_3 \in \mathcal{K}_3$ and ax .
- $I_3.\phi(\text{sk}_3)$. For $\text{sk}_3 = (\overline{\mathbf{b}}, \overline{\mathbf{c}}) \in (\mathbb{Z}_q^{k+1})^2$, outputs $\text{pk}_3 = ([\overline{\mathbf{Pb}}], [\overline{\mathbf{Pc}}])$.
- $I_3.\text{Priv}(\text{sk}_3, x, \tau)$. For $\text{sk}_3 = (\overline{\mathbf{b}}, \overline{\mathbf{c}}) \in (\mathbb{Z}_q^{k+1})^2$, $x = [\overline{\mathbf{x}}^\top]$ and $\tau \in \overline{\mathbb{G}}$, outputs hash value $\pi = (\pi_1, \pi_2) = \left([(\overline{\mathbf{x}}^\top + \overline{\lambda}_1^\top \overline{\mathbf{P}})\tau\overline{\mathbf{b}}], [(\overline{\mathbf{x}}^\top + \overline{\lambda}_2^\top \overline{\mathbf{P}})\tau\overline{\mathbf{c}}] \right)$.
- $I_3.\text{Pub}(\text{pk}_3, x, w, \tau)$. For $\text{pk}_3 = ([\overline{\mathbf{Pb}}], [\overline{\mathbf{Pc}}])$, $x = [\mathbf{w}^\top \overline{\mathbf{P}}]$ with witness \mathbf{w} and $\tau \in \overline{\mathbb{G}}$, outputs $\pi = (\pi_1, \pi_2) = \left([(\mathbf{w}^\top + \overline{\lambda}_1^\top)\tau\overline{\mathbf{Pb}}], [(\mathbf{w}^\top + \overline{\lambda}_2^\top)\tau\overline{\mathbf{Pc}}] \right)$.
- $I_3.\text{RandX}(x, x', r_x)$. For $x = [\overline{\mathbf{x}}^\top]$, $x' = [\overline{\mathbf{x}}'^\top] \in \overline{\mathcal{X}}$ and $r_x \in \mathbb{Z}_q$, outputs $x^* = [\overline{\mathbf{x}}^\top + r_x \overline{\mathbf{x}}'^\top]$.
- $I_3.\text{RandT}(\tau, r_\tau)$. For $\tau \in \overline{\mathbb{G}}$ and $r_\tau \in \mathbb{Z}_q$, outputs $\tau^* = r_\tau \cdot \tau$.
- $I_3.\text{RandH}(\pi, \pi', r_x, r_\tau)$. For $\pi = (\pi_1, \pi_2)$, $\pi' = (\pi'_1, \pi'_2)$, $r_x \in \mathbb{Z}_q$ and $r_\tau \in \mathbb{Z}_q$, outputs $\pi^* = ((\pi_1 \cdot (\pi'_1)^{r_x})^{r_\tau}, (\pi_2 \cdot (\pi'_2)^{r_x})^{r_\tau})$.

As for I_4 , its algorithms $I_4.\phi$, $I_4.\text{RandX}$, $I_4.\text{RandT}$ and $I_4.\text{RandH}$ are the same as $I_3.\phi$, $I_3.\text{RandX}$, $I_3.\text{RandT}$ and $I_3.\text{RandH}$. Besides, $I_4.\text{Setup}$ is the same as $I_3.\text{Setup}$ except that ax is null and the hash function family is $\widetilde{H}_{(\cdot)} : \overline{\mathcal{X}} \times \mathcal{T}_4 \rightarrow \overline{\mathbb{G}}^2$ where $\mathcal{T}_4 = \mathcal{T}_3$. $I_4.\text{Priv}$ and $I_4.\text{Pub}$ are equivalent to $I_3.\text{Priv}$ and $I_3.\text{Pub}$ with $\overline{\lambda}_1 = \overline{\lambda}_2 = \mathbf{0}$.

- $I_4.\text{Priv}(\text{sk}_4, x, \tau)$. For $\text{sk}_4 = (\overline{\mathbf{b}}, \overline{\mathbf{c}}) \in (\mathbb{Z}_q^{k+1})^2$, $x = [\overline{\mathbf{x}}^\top] \in \overline{\mathcal{X}}$ and $\tau \in \overline{\mathbb{G}}$, outputs $\pi = (\pi_1, \pi_2) = ([\overline{\mathbf{x}}^\top \tau \overline{\mathbf{b}}], [\overline{\mathbf{x}}^\top \tau \overline{\mathbf{c}}])$.
- $I_4.\text{Pub}(\text{pk}_4, x, w, \tau)$. For $\text{pk}_4 = ([\overline{\mathbf{Pb}}], [\overline{\mathbf{Pc}}])$, $x = [\mathbf{w}^\top \overline{\mathbf{P}}]$ with witness \mathbf{w} and $\tau \in \overline{\mathbb{G}}$, outputs $\pi = (\pi_1, \pi_2) = ([\mathbf{w}^\top \tau \overline{\mathbf{Pb}}], [\mathbf{w}^\top \tau \overline{\mathbf{Pc}}])$.

One can verify the correctness and 1-smoothness of I_3 and I_4 . Analogous to the proofs of Theorem 8, 9, 10, 11 and 12, one can easily prove that if $k \geq 2$, I_3 is perfectly pairwise-rerandomizable, PT-Smooth₁ and CPR-Smooth with respect to I_4 , and I_4 is perfectly self-rerandomizable and CSR-Smooth. The concrete proofs are given in the full version [29].

6.3 Concrete PKE from k -LIN Assumption

Figure 6 depicts the full concrete scheme PKE based on k -LIN assumption. Note that the group $\overline{\mathbb{G}}$ and \mathbb{G} should be chosen relevantly to ensure that u in tag τ could be encrypted with proper group. Concretely, let $\overline{\mathbb{G}} = \mathbb{QR}_{2q+1}^*$ and $\mathbb{G} = \mathbb{QR}_{2p+1}^*$ be two groups of quadratic residues where $p = 2q+1$ and $(q, 2q+1, 4q+3)$ is a sequence of primes, called a Cunningham chain (of the first kind) of length 3.

Acknowledgement. We would like to thank all anonymous reviewers for their valuable comments. This work is supported in part by the National Natural Science Foundation of China (Grant No. 62032005, Grant No. 61702541, and Grant No. 61872087), Science Foundation of Fujian Provincial Science and Technology Agency (Grant No. 2020J02016).

References

1. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_6
2. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
3. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHF and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_25
4. Benhamouda, F., Bourse, F., Lipmaa, H.: CCA-secure inner-product functional encryption from projective hash functions. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 36–66. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_2
5. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_33
6. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 281–300. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_18
7. Chen, R., Mu, Y., Yang, G., Susilo, W., Guo, F., Zhang, M.: Cryptographic reverse firewall via malleable smooth projective hash functions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 844–876. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_31
8. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
9. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls—secure communication on corrupted machines. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 341–372. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_13
10. Faonio, A., Fiore, D.: Improving the efficiency of re-randomizable and replayable CCA secure public key encryption. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 2020. LNCS, vol. 12146, pp. 271–291. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57808-4_14
11. Faonio, A., Fiore, D., Herranz, J., Ràfols, C.: Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 159–190. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_6

12. Ganesh, C., Magri, B., Venturi, D.: Cryptographic reverse firewalls for interactive proof systems. In: 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020). <https://doi.org/10.4230/LIPIcs.ICALP.2020.55>
13. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal re-encryption for mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24660-2_14
14. Groth, J.: Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 152–170. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_9
15. Han, S., Liu, S., Lyu, L., Gu, D.: Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 417–447. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_15
16. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: how secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_19
17. Libert, B., Peters, T., Qian, C.: Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 247–276. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_11
18. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 657–686. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_22
19. Peng, K., Nieto, J.M., Desmedt, Y., Dawson, E.: Klein bottle routing: an alternative to onion routing and mix network. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 296–309. Springer, Heidelberg (2006). https://doi.org/10.1007/11927587_25
20. Pereira, O., Rivest, R.L.: Marked mix-nets. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 353–369. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_22
21. Phan, D.H., Pointcheval, D.: OAEP 3-round: a generic and secure asymmetric encryption padding. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 63–77. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30539-2_5
22. Prabhakaran, M., Rosulek, M.: Rerandomizable RCCA encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 517–534. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_29
23. Saito, J., Ryou, J.-C., Sakurai, K.: Enhancing privacy of universal re-encryption scheme for RFID tags. In: Yang, L.T., Guo, M., Gao, G.R., Jha, N.K. (eds.) EUC 2004. LNCS, vol. 3207, pp. 879–890. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30121-9_84
24. Senftleben, M., Bucicou, M., Tews, E., Armknecht, F., Katzenbeisser, S., Sadeghi, A.-R.: MoP-2-MoP – mobile private microblogging. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 384–396. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_25
25. Shoup, V.: A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112 (2001). <http://eprint.iacr.org/2001/112>
26. Syverson, P., Dingledine, R., Mathewson, N.: Tor: The second generation onion router. In: Usenix Security (2004). https://www.usenix.org/legacy/events/sec04/tech/full_papers/dingledine/dingledine.html/

27. Wee, H.: KDM-security via homomorphic smooth projective hashing. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016, Part II. LNCS, vol. 9615, pp. 159–179. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_7
28. Young, A.L., Yung, M.: Semantically secure anonymity: foundations of re-encryption. In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 255–273. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_14
29. Wang, Y., Chen, R., Yang, G., Huang, X., Wang, B., Yung, M.: Receiver-anonymity in rerandomizable RCCA-secure cryptosystems resolved. Cryptology ePrint Archive, Report 2021/862 (2021). <http://eprint.iacr.org/2021/862>

Foundations



White Box Traitor Tracing

Mark Zhandry^{1,2}(✉)

¹ Princeton University, Princeton, USA

² NTT Research, Sunnyvale, USA

mark.zhandry@ntt-research.com

Abstract. Traitor tracing aims to identify the source of leaked decryption keys. Since the “traitor” can try to hide their key within obfuscated code in order to evade tracing, the tracing algorithm should work for general, potentially obfuscated, decoder *programs*. In the setting of such general decoder programs, prior work uses *black box* tracing: the tracing algorithm ignores the implementation of the decoder, and instead traces just by making queries to the decoder and observing the outputs.

We observe that, in some settings, such black box tracing leads to consistency and user privacy issues. On the other hand, these issues do not appear inherent to *white box* tracing, where the tracing algorithm actually inspects the decoder implementation. We therefore develop new white box traitor tracing schemes providing consistency and/or privacy. Our schemes can be instantiated under various assumptions ranging from public key encryption and NIZKs to indistinguishability obfuscation, with different trade-offs. To the best of our knowledge, ours is the first work to consider white box tracing in the general decoder setting.

1 Introduction

Traitor tracing [CFN94] deters piracy by embedding identifying information into users’ personalized decryption keys. From a leaked key, it should be possible to “trace,” extracting the “traitor’s” identifying information; with this information, remedial action can be taken such as fines, prosecution, and/or revocation. Tracing is ideally possible even in a variety of adversarial scenarios, such as if many users collude or if the secret key is hidden inside an obfuscated decoder program. The bulk of the tracing literature has focused on reducing the sizes of various components, such as ciphertexts and public and secret keys.

Analyzing (potentially obfuscated) program code is notoriously difficult. Consequently, most recent traitor tracing works (e.g. [CFN94,BSW06,BN08,BZ14,NWZ16,GKW18,Zha20]) operate in a *black box* model: the tracer actually does *not* try to inspect the particular software of the decoder, but instead simply queries the decoder on various ciphertexts and observes the outputs. From just the input/output behavior, the tracer is able to extract the identifying information¹.

¹ Another oft-cited reason to consider black box tracing is that the decoder could be contained in a hardware device employing various tamper resistant mechanisms to prevent its code from being inspected. In this work, however, we will only consider software decoders.

This work: white box tracing. In this work, we consider the use of *white box* algorithms for tracing general decoders. Specifically, we allow the adversary to produce arbitrary (potentially obfuscated) programs in an attempt to remove the embedded identifying information, but use non-black box algorithms for tracing. The two main questions we explore in this work are:

*What are potential advantages of white box tracing? And,
How to white box trace general adversarial decoders?*

Remark 1. An early model for traitor tracing, which we will call *faked key* tracing, stipulates that the traitor outputs an actual valid key for the system. Tracing then uses the combinatorial or algebraic structure of the key, as opposed to its input/output behavior. Many early traitor tracing works consider faked key tracing [KD98, BF99, NP01, KY03, TS06, JKL09, JKL09, ADVW13], and some refer to this model as “white box tracing” or “non-black box tracing” (see, e.g. [NDC+15] and [GNPT13], respectively). Such naming reflects that non-black box tracing algorithms have always coincided with tracing models where the traitor must output a valid key. Outside of traitor tracing, however, the labels “white box” or “non-black box” refer to the type of access to a program, and is potentially orthogonal to the format of the program. We therefore prefer the terms “faked key” versus “general decoder” to refer to the structure of the adversary’s decoder, and terms “black box” versus “white box” to refer to the level of access the tracing algorithm has to the decoder. To the best of our knowledge, ours is the first work exploring white box tracing in the general decoder setting.

1.1 Motivation

To motivate white box tracing, we now discuss limitations of black box tracing; overcoming these limitations will be the focus of our work. These limitations are orthogonal to the “usual” goal of traitor tracing, namely minimizing parameter sizes. As such, parameter sizes are only a secondary consideration in this work.

Public tracing. We first motivate a particular type of traitor tracing which has both *public tracing* and *embedded identities*. Embedded identities, originally proposed by Nishimaki et al. [NWZ16], means that arbitrary information can be embedded in the secret keys; in contrast, most tracing schemes only embed an index from a polynomial-sized set. Nishimaki et al. point out that the tracer would naturally want to know useful identifying information about the traitor, in order to prosecute or fine. The key issuer could of course maintain a database mapping user indices to actual identifying information, but having to store such a database in the clear naturally creates privacy concerns. Embedded identities allow this information to be stored directly in the issued keys themselves, eschewing the need for such a database.

For public tracing, the tracing algorithm only needs the public key and no secrets. This is in contrast to *secret tracing*, where a secret key is required to

trace, and anyone with the secret key can break the security of the system. There are at least a few reasons to prefer public tracing algorithms:

- Secret key tracing means the tracer cannot be compromised. Public key tracing allows *anyone* to trace, removing a potential point of failure.
- As explained in [Pfi96], private tracing provides no natural mechanism for submitting evidence to a judge, as there is no way besides revealing the secret tracing key to certify the results of tracing. While there are solutions in the secret tracing setting, public tracing automatically solves the problem: the judge can always verify by simply re-tracing with the public key.
- In private key tracing, the tracer must somehow discover the decoder in order to trace, and deterrence therefore relies on such discovery. It may take time for the tracer to discover the decoder program, or it may never be discovered if the traitor and an unauthorized user are secretive enough in their communication. After all, the pirate decoder would naturally be transmitted out of band, and there is no reason to believe the tracer would automatically see the decoder.

In contrast, with public key tracing, the sensitive information is immediately revealed to *anyone* who receives the decoder, including the un-authorized user. Especially when combined with embedded identities, public tracing yields a very strong deterrent mechanism: for example, if the embedded information contains a bank account number, then a traitor would likely be unwilling to send their key to anyone else, especially the unscrupulous un-authorized user.

The good news is that there already exist such public embedded identity tracing schemes, with different trade-offs in terms of parameter sizes and assumptions used [NWZ16, GKW19]. However, as we will now explain, public *black box* tracing schemes, including all existing public tracing schemes for general decoders, are inherently vulnerable to certain kinds of attacks.

Problem 1: Privacy. Consider an encrypted group chat application, where a group of users broadcast messages to the entire group. The broadcasts are encrypted to protect against eavesdropping. The group members are also mutually distrusting, and want to protect against a traitor revealing their key to an outside user. For example, the group could consist of political dissidents coordinating a protest against an authoritarian regime, and they are concerned that a member may give their key to government agents.

The group therefore will use a traitor tracing scheme to encrypt their message, embedding sensitive or identifying information of each user into personalized decryption keys. In this decentralized scenario, it is unclear who the tracer should be, and also unclear how to securely maintain a database of users' identifying information. Therefore, the group would naturally want a scheme with public tracing and embedded identities, as discussed above².

² Similar to [NWZ16], we envision the original setup and key distribution executed through multiparty computation, so no single user is responsible for setup.

Unfortunately, we observe that such constructions inherently come with privacy concerns, due to having black box tracing. A malicious Alice (whether or not an authorized group member) may try to steal Bob’s private information by running the tracing algorithm over the network. That is, Alice can send messages to Bob, and see how he responds, mounting a chosen ciphertext attack against Bob’s decryption functionality. This is exactly how black box tracing works, and since the scheme has public tracing, Alice has all the access she needs to trace Bob’s key. In fact, tracing algorithms typically work in the “minimal access” setting, meaning Alice only needs to know whether Bob decrypts. In our political dissident scenario above, this means government agents may be able to learn the identities of the dissidents (or whatever sensitive information is embedded in the user keys) by simply posting messages to the group chat, and seeing if there are any responses. This would naturally concern the group members.

Thus, it is impossible to get the best of all worlds—public tracing, embedded sensitive information, and user privacy under chosen ciphertext attacks—with black box tracing. White box tracing, on the other hand, may offer a solution: the remote attacker never actually sees the user’s decryption program, and may therefore be unable to run a white box tracing algorithm. Of course, at this point it is not clear how to actually use white box tracing to achieve these goals.

Remark 2. Nishimaki et al. consider an object they call “anonymous” traitor tracing, where the users never reveal their identifying information to the key issuer. However, beyond motivating the direct embedding of sensitive information within user keys, they do not further explore anonymity or privacy in traitor tracing. In particular, they do not discuss the privacy issue we observe here.

Problem 2: Consistency. In traitor tracing, the functionality of the various user keys is different. This is inherent, for similar reasons to the case of watermarking as explained by Barak et al. [BGI+01]: if each user key had identical functionality, then a traitor could apply *indistinguishability obfuscation* (iO) to their key. The guarantees of iO would then imply the obfuscations of different users’ keys are computationally indistinguishable, and hence cannot be efficiently traced. Thus any efficient tracing scheme that maintains perfect consistency would necessarily prove the non-existence of iO, which currently seems out of reach.

This means certain ciphertexts will decrypt differently under different user keys. While these differing inputs would not occur under normal operation, it is nonetheless easy to find differing inputs in the case of *public* black box tracing: since the tracing algorithm must distinguish between the keys by querying the decoder functionality, it must be querying exactly on these differing inputs.

To see why this might be an issue, consider executing a multi-party computation (MPC) protocol. As is standard in the MPC literature, assume that the users have access to a reliable broadcast channel: when one user sends a message on the channel, all other users are guaranteed to receive the same message. Now, suppose that the set of users want to encrypt the broadcasts in their MPC

protocol using a traitor tracing scheme³. Unfortunately, even if the ciphertexts are sent over a reliable broadcast channel, the MPC is run on a virtual plaintext channel that is *not* reliable: a malicious user may broadcast a ciphertext specifically designed to decrypt differently by different users. Typical MPC protocols require broadcasts to be received consistently between the various users, and such an inconsistent decryption would break the guarantees of the MPC protocol⁴.

More generally, in any setting where a broadcast channel is needed to ensure that all users receive the same message, encrypting the communication with a public black box tracing scheme can result in an unreliable broadcast channel.

On the other hand, such consistency issues do not appear inherent to white box tracing: a malicious user only has black box access to the other users' decryption functionalities, and may be unable to use this access to find an input that decrypts differently. Yet if a user actually leaks their decryption key or an obfuscated decoder with the key inside, a white box tracing algorithm might nevertheless be able to trace, and in particular may be able to find such a differing ciphertext by inspecting the code. Again, it is not yet clear how exactly to use white box tracing to achieve this goal.

1.2 Overview of Our Results

In this work, we give several new results for white box traitor tracing:

- In Sect. 3, we give definitions for privacy and consistency in the traitor tracing setting. Formalizing the above observations, we show that it is impossible to satisfy either privacy or consistency while simultaneously achieving the tracing guarantee with a public black box tracing algorithm.
- In Sect. 5, we construct a secure white box public tracing system that also satisfies our privacy notion. Our construction can be based on either generic public key encryption and non-interactive zero knowledge, or on indistinguishability obfuscation (iO), with different trade-offs in terms of collusion resistance and parameter sizes. This scheme is not consistent.
- We do not fully solve the consistency problem, but in Sect. 6 we demonstrate a white box traitor tracing scheme that achieves the tracing guarantee for *constant*-sized collusions, while also achieving consistency and privacy (both for arbitrary collusions). Our scheme uses fully homomorphic encryption, compute-and-compare obfuscation, and non-interactive zero knowledge, which are all implied by circularly secure Learning With Errors (LWE) or iO.

Along the way, we introduce and build a notion of black box function privacy for functional encryption (Sect. 4), which may be of independent interest.

³ Of course, MPC security already implies that outsiders will be unable to learn anything about the users' inputs even without encrypting. But perhaps the users are encrypting messages for other reasons.

⁴ There are MPC protocols that do not need broadcast channels, but they often come at the cost of increased round complexity (see [CGZ20] and references therein).

1.3 Future Directions

Our work motivates several fascinating future directions:

- We are able to construct consistent tracing only for constant-sized collusions. Is arbitrary-collusion consistent tracing possible, potentially even under extremely strong assumptions? Alternatively, is there an impossibility?
- Our constructions utilize heavy machinery, using non-black box techniques at many levels. This leads our constructions to be inefficient. Can truly efficient white box tracing be achieved?
- Traitor tracing can be seen as a special case of the more general problem of software watermarking. To the best of our knowledge, all works in the watermarking setting also use black box mark detection/extraction, and the privacy and consistency issues naturally translate to watermarking. Can white box techniques be used to overcome similar issues in watermarking?
- There is a large gap between known programs that can be watermarked (e.g. puncturable PRFs [CHN+16]) and programs that are known to be un-watermarkable (e.g. un-obfuscatable functions [BGI+01]). Can white box mark detection/extraction be used to help close this gap?
- Chosen ciphertext attacks for traitor tracing has been considered before (e.g. [DF03]), but to the best of our knowledge, prior such explorations have been limited to message secrecy goals. Our work highlights further consequences of CCA attacks. Are there other possible consequences?

2 Our Techniques

2.1 Part 1: Private Traitor Tracing

Definitions and impossibility. In Sect. 3.2, we give a formal model for privacy of a user’s sensitive information under chosen ciphertext queries to their decryption functionality. We give several indistinguishability and simulation-based definitions formalizing “learning nothing”; we show that the indistinguishability notions are equivalent to the corresponding simulation-based notions. Our strongest notion actually gives the adversary the full master secret key, meaning privacy holds even if the master key is leaked. We also formalize the above observations, showing that even the weakest versions of our privacy notion are impossible for black box public tracing schemes. Our strongest notion of privacy (where the adversary gets the master secret key) is even incompatible with black box *secret* tracing.

Theorem 1 (Informal). *Black box publicly traceable schemes cannot be private.*

Achieving privacy through white box tracing. In Sect. 5 turn to building a scheme that can be publicly traced while maintaining privacy. Since black box tracing is impossible, we must devise a tracing scheme that is inherently white box, in that accessing the code of decoder allows for tracing while black box access cannot.

The natural starting point are the un-obfuscatable functions (UOFs) of Barak et al. [BGI+01], which can be learned from *any* code for the function, but not from black box access. While UOFs are indeed closely related to our goal, they do not immediately give what we need:

- UOFs are not necessarily decryption functions. While Barak et al. show how to extend their UOFs to *encryption* functions, it is not obvious that they can be extended to the *decryption* functionality.
- In Barak et al.’s UOF, the functionality of the encryption scheme is tied to the UOF itself. In the traitor tracing system, we want many different users to be able to decrypt the same ciphertext, hence seemingly all keys would have the same UOF. But at the same time, the users should have different sensitive information, seemingly requiring different UOFs.
- Barak et al.’s UOF does not handle the case of colluding users.
- Barak et al.’s UOF requires perfect or near-perfect correctness for the decoder. While this can be extended to much lower correctness using *robust* un-obfuscatable function [BP13], the current techniques do not extend to the inverse-polynomial correctness setting, as usually required in traitor tracing.

Our idea is to use *black box* traitor tracing techniques, but set the embedded information for a user to be a UOF in order to upgrade the black box scheme into a white box scheme. However, this requires care. The naive approach is to set the embedded information to be the actual code of the UOF, but this will not work: the adversary can mount the black box tracing algorithm remotely to recover the UOF code, and then recover the user’s private information from the UOF code. We therefore need a more sophisticated embedding.

To describe our solution, we first recall the black box tracing scheme of Nishimaki et al. [NWZ16], which follows the PLBE framework [BSW06]. Start from a *public key functional encryption* (FE), which allows for generating secret keys sk_f for functions g ; sk_f allows for learning $f(x)$ from an encryption of x , but nothing else about x . Assume the identity space is $[I]$, for some exponentially-large integer I . Nishimaki et al. encrypt a message m by FE-encrypting the pair $(0, m)$. The secret key with identity id embedded is then $\text{sk}_{f_{\text{id}}}$ where

$$f_{\text{id}}(z, m) = \begin{cases} m & \text{if } \text{id} > z \\ \perp & \text{if } \text{id} \leq z \end{cases} .$$

Notice that $f_{\text{id}}(0, m) = m$, meaning $\text{sk}_{f_{\text{id}}}$ will decrypt honest ciphertexts, while $f_{\text{id}}(I, m) = \perp$. By FE security, given any decoder D built from $\text{sk}_{f_{\text{id}}}$ and any z , one can test if $\text{id} > z$ by looking at the decoder’s decryption probability on encryptions of (z, m) . A binary search over z can then recover id ; Nishimaki et al. show how to extend the binary search to the case of colluding users and to decoders with small decryption probability, as required for traitor tracing.

In order to embed a *function* into the secret key, rather than just a string, we modify Nishimaki et al.’s construction as follows. To embed a function g , we choose a random “tag” τ , and generate the function secret key $\text{sk}_{f_{g, \tau}}$ where

$$f_{g,\tau}(z, x, m) = \begin{cases} m & \text{if } \tau > z \\ \perp & \text{if } (\tau < z) \vee (\tau = z \wedge x = \perp) \\ m & \text{if } \tau = z \wedge x \neq \perp \wedge g(x) = 1 \\ \perp & \text{if } \tau = z \wedge x \neq \perp \wedge g(x) = 0 \end{cases}.$$

If we set $x = \perp$, then $\text{sk}_{f_{g,\tau}}$ has the same structure as f_{id} above with $\text{id} = \tau$. Therefore, we can first run a binary search over z to recover τ . Then we evaluate g on an input x by testing the decoder on encryptions of (τ, x, m) , and seeing whether it is able to decrypt; if it can decrypt, we must have $g(x) = 1$, otherwise $g(x) = 0$. The result is what we call *function-embedded* traitor tracing (FETT).

Remark 3. The structure above is similar to an optimization Nishimaki et al. employ to get a scheme where ciphertexts are very short, in particular shorter than the identity id . Essentially, they let g be the function with polynomial-sized domain whose truth table is id . The structure was also used in [GKW19], again with g being a truth table, with the goal of achieving efficient traitor tracing under standard assumptions. Our use of this structure is with an entirely different goal: to embedding functions in a non-trivial way into the secret keys.

With a FETT, we can now build our private traitor tracing scheme by setting g to be an un-obfuscatable function UOF_{id} , which has the identity id of the user embedded. Given a decoder D , we can construct a program P that evaluates UOF_{id} by running the FETT tracing algorithm as described above. The un-obfuscatable function guarantee means that from P , we can extract the identity id . Meanwhile, the intuition for privacy is that a remote user can only make black box queries to the user, corresponding to black box queries to UOF_{id} ; the un-obfuscatable function guarantee means that such queries do not reveal id . Realizing this intuition, however, comes with several challenges:

- Since tracing is randomized, the program P is a randomized procedure, whereas the un-obfuscatable function guarantee of Barak et al. only applied to deterministic circuits. One can make P deterministic by hard-coding the randomness, but for any particular choice of randomness there may be some x where P outputs the incorrect answer. Additionally, the original P may actually completely fail to evaluate UOF_{id} correctly on some inputs, for example if x is the master secret key. Fortunately, we show that P correctly computes UOF_{id} for inputs x that are efficiently computable to the adversary. We show that the Barak et al. functions maintain the un-obfuscatable function guarantee even for this relaxed notion of correctness for P^5 .
- For a secure FE, black box queries to the secret key sk_f might still reveal the code for f . If such an FE is used in our construction, the result is that a remote adversary may be able to obtain the code for UOF_{id} , and hence id . As such, we actually need a function privacy notion for the FE scheme, which

⁵ Our needed notion is also implied by *robust* UOFs [BP13], but these are only known from trapdoor permutations. Barak et al.’s construction relies on just one-way functions, which are implied by FE.

roughly requires that black box queries to sk_f are “no better than” black box queries to f itself. This notion of function privacy is incompatible with existing notions of privacy for public key functional encryption⁶. In Sect. 4, we show a simple transformation upgrading any plain FE scheme into one with our notion of function privacy using non-interactive zero knowledge (NIZK) proofs.

Remark 4. Our “black box” function privacy notion may have applications beyond this work. For example, consider a common-cited application of functional encryption to filtering spam. Here, f is a spam filter employed by an email server. A user wants the server to be able to route encrypted emails according to the spam filter, but does not want the server to learn anything beyond whether or not an email was spam. The solution is to give the server sk_f . But now suppose that f is proprietary, and the server wants to prevent potential spammers from learning too much about f ⁷. A spammer can effectively query f by sending spam to the user and seeing whether or not the user actually receives the email (as indicated, say, by whether the user clicks on a link). Plain functional encryption, unfortunately, may allow the adversary to do more: the result of decrypting malicious ciphertexts may reveal the bits of sk_f , or even the code of f . Black box function privacy guarantees that the spammer is limited to just querying f and learning the input/output behavior of f .

Instantiations. We can plug any FE scheme (and NIZK) into our construction. Our conversions preserve the ciphertext sizes of the underlying FE. Using known FE constructions, we obtain the following:

Theorem 2 (Informal). *Assuming public key encryption and NIZKs, there exists a traitor tracing scheme with public tracing, embedded identities, and privacy, with $\text{poly}(N)$ -sized ciphertexts for collusions of size N . Assuming indistinguishability obfuscation and one-way functions, there exists such a scheme with $O(1)$ -sized ciphertexts.*

2.2 Part 2: Toward Consistent Traitor Tracing

Next, we turn to the problem of making sure different users decrypt consistently.

Definition and impossibility. In Sect. 3.3, we define several variants of consistency. The strongest requires that even if the master secret key is leaked, it is impossible to find a ciphertext that decrypts differently under any two users. This variant is quite strong, and we do not know how to achieve it. Instead we also consider weaker variants. The variant we ultimately achieve requires that a malicious user, or group of users, cannot find a ciphertext that would cause

⁶ Full function privacy, which in particular implies black box function privacy, is possible in the *secret key* setting for functional encryption [BS15].

⁷ To prevent the user himself from learning f , we can imagine sk_f is generated using a multiparty computation protocol between the server and user.

two *honest* users to decrypt differently. Note that our notion *does* allow a group of malicious users to find ciphertexts that *they* decrypt differently (or that they decrypt differently than the honest user(s)). We nevertheless believe our notion is meaningful, as the consistency between honest users seems most important. We formalize the above observations, showing that even the weakest versions of our consistency notion are impossible for black box public tracing schemes. Our strongest notion of consistency (where the adversary gets the master secret key) is even incompatible with black box *secret* tracing schemes.

Theorem 3 (Informal). *Black box publicly traceable schemes cannot be consistent.*

Challenges. We first observe that our private traitor tracing scheme is *not* consistent, a consequence of the first step of our tracing algorithm being black box. First, known UOFs for circuits are non-evasive, with the accepting inputs depending on the function. With this fact, the black box tracing step can easily be used to find differing inputs. An even more basic reason is that tracing recovers the tags τ , and keys with different tags have different functionalities. For privacy, these are not concerning since the tags and non-evasive parts of the UOFs are independent of the identifying information that must be kept secret. For consistency, however, these issues mean it is easy to find differing inputs. Even if one can find evasive UOFs for circuits, the tag problem will persist, as secret keys must have distinct tags for collusion-resistance.

Our Construction. We are unable to fully solve the consistency problem. However, in Sect. 6 we achieve a solution which remains traceable with public tracing for *constant-sized* collusions, and achieves consistency (and privacy) for arbitrary collusions.

At a very high level, our idea is to restructure $f_{g,\tau}$ to require a special key σ in order to activate the functionality g . By keeping σ secret, we can guarantee that differing inputs cannot be found. However, keeping σ secret means tracing is no longer possible. To overcome this issue, we encrypt σ using a fully homomorphic encryption (FHE) scheme. We can then run the tracing algorithm homomorphically on the encryption of σ , arriving at an encryption of the users' sensitive information. Of course, we now need a way to decrypt to recover the information in the clear, without using the FHE decryption key (recall that we want *public* tracing). We show that, by providing a certain compute-and-compare obfuscation in the public key [WZ17, GKW17], we can allow for decrypting in exactly the instance where tracing succeeds.

Unfortunately, the above is not consistent if the adversary has even a single key. This is because any user with a secret key can run the tracing algorithm on their key. It is not difficult to show that doing so will actually allow the user to decrypt any FHE ciphertext—including recovering σ —bring us back to square one. The natural solution is to have different σ and different FHE instances for each user, so that a user can recover their own σ but no one else's. But if the σ are isolated in different instances, there is no way to simultaneously provide the σ for different users when homomorphically running the tracing algorithm.

Our solution is to provide a unique σ and FHE instance for each *subset* of users; we set σ to simply be a signature on the (description of the) set. Of course, there are exponentially-many such subsets, so we only consider subsets of a *constant* size c to keep the number of subsets polynomial. Tracing then runs homomorphically on every subset of tags, using the compute-and-compare obfuscations to check if tracing succeeded, and if so recovering the sensitive information of the users for that subset. We prove that, as long as at most c users collude, at least one of the subsets will succeed during honest tracing.

Now, an adversary controlling a set S of secret keys will be able to run tracing on any subset of S , and would be able to find the σ for that subset. Using such σ would allow the adversary to find inputs that differ amongst the keys they control. However, we show that the σ for any set *not* entirely contained in S remains hidden. This is sufficient to show that the adversary cannot find differing inputs for the honest users.

By instantiating our scheme with known FHE and compute-and-compare obfuscations, we obtain the following:

Theorem 4 (Informal). *Assuming circularly secure learning with errors, or both sub-exponentially secure indistinguishability obfuscation and lossy encryption, for any constant c , there exists a traitor tracing scheme with public tracing, embedded identities, and consistency, tolerating c collusions.*

3 Traitor Tracing Definitions

Here, we define traitor tracing, including our new notions of privacy and consistency. Our actual constructions will be given in Sects. 4, 5, and 6.

3.1 Basic Tracing Definition

We first recall the definition of (plain) traitor tracing appearing in recent works [NWZ16, GKRW18, GKW18, Zha20]. A traitor tracing scheme is a tuple $\Pi_{\text{TT}} = (\text{Gen}, \text{Enc}, \text{Derive}, \text{Dec}, \text{Trace})$ of PPT algorithms:⁸

$$\begin{aligned} (\text{pk}, \text{msk}) &\leftarrow \text{Gen}(1^\lambda, N) & \text{sk} &\leftarrow \text{Derive}(\text{msk}, \text{id}) & m &\leftarrow \text{Dec}(\text{sk}, c) \\ c &\leftarrow \text{Enc}(\text{pk}, m) & A &\leftarrow \text{Trace}(\text{pk}, D, m_0, m_1, 1^N, 1^{1/\epsilon}) . \end{aligned}$$

Above, λ is the security parameter, N an upper bound on the number of users, pk the public key, msk the master secret key, id a user identity, sk a user-specific secret key, m a message, and c a ciphertext, D the code of a decoder program, m_0, m_1 two challenge messages, and $\epsilon \in (0, 1/2]$ a “goodness” parameter. We require that there exists a negligible function negl such that for all $\lambda > 0, N > 0, \text{id}, m$:

$$\Pr \left[\text{Dec}(\text{sk}_{\text{id}}, c) = m : \begin{array}{l} (\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{sk}_{\text{id}} \leftarrow \text{Derive}(\text{msk}, \text{id}) \\ c \leftarrow \text{Enc}(\text{pk}, m) \end{array} \right] \geq 1 - \text{negl}(\lambda) .$$

Consider the following experiment on adversary \mathcal{A} and parameter $\epsilon = \epsilon(\lambda)$:

⁸ Note that N may be allowed to be super-polynomial.

- \mathcal{A} gets input 1^λ , and produces a number N .
- Run $(\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N)$. Send pk to \mathcal{A} .
- \mathcal{A} then makes at most N “identity” queries on identities id . For each query, respond with $\text{sk}_{\text{id}} \leftarrow \text{Derive}(\text{msk}, \text{id})$. Let T be the set of id queried.
- \mathcal{A} outputs D and m_0, m_1 . Run $A \leftarrow \text{Trace}(\text{pk}, D, m_0, m_1, 1^{|T|}, 1^{1/\epsilon})$.

We define the following events. $\text{BadTr}_\epsilon(\mathcal{A}, \lambda)$ means an honest user is accused: $A \notin T$. $\text{GoodDec}_\epsilon(\mathcal{A}, \lambda)$ means the decoder succeeds in distinguishing encryptions of m_0 and m_1 : $\Pr[D(c) = b : b \leftarrow \{0, 1\}, c \leftarrow \text{Enc}(\text{pk}, m_b)] \geq 1/2 + \epsilon(\lambda)$. In this case, we say D is “good.” $\text{GoodTr}_\epsilon(\mathcal{A}, \lambda)$ means *someone* is accused: $|A| > 0$.

Definition 1. A traitor tracing scheme Π_{TT} is traceable if, for all PPT \mathcal{A} and inverse-poly ϵ , there exists a negligible function negl such that $\Pr[\text{BadTr}_\epsilon(\mathcal{A}, \lambda)] \leq \text{negl}(\lambda)$ and $\Pr[\text{GoodTr}_\epsilon(\mathcal{A}, \lambda)] \geq \Pr[\text{GoodDec}_\epsilon(\mathcal{A}, \lambda)] - \text{negl}(\lambda)$.

We will occasionally distinguish between traitor tracing schemes where Trace has full access to the code of D versus schemes where Trace only makes queries to the decoder. We will say that a scheme where Trace has full access is *white box traceable*, and a scheme where Trace only makes queries is *black box traceable*; for the latter we write $\text{Trace}^D(\text{pk}, m_0, m_1, 1^N, 1^{1/\epsilon})$. We note that most prior work on traitor tracing explicitly defines tracing to be black box.

We will also consider traitor tracing with bounded collusions, where N is bounded to some value c , which may be a function of λ . In this case, we say the scheme is *c-bounded collusion traceable* (or *c-traceable*, for short).

Remark 5. The bulk of the tracing literature sets the identity space to be $[N]$. Starting with Nishimaki et al. [NWZ16], some recent works have considered the case where the identity space is exponentially large, say n -bit strings. These works often use terminology such as “embedded identities” [GKW19] to disambiguate from the usual setting. In this work, we will largely ignore such distinctions.

3.2 Private Traitor Tracing

We now give our new definition of privacy in traitor tracing. Let \mathcal{A} be an adversary, and consider the following experiment:

- \mathcal{A} gets input 1^λ , and produces a number N .
- Run $(\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N)$, and send pk to \mathcal{A} . \mathcal{A} can now make two kinds of queries, in any order:
 - At most N “identity” queries on identities id ; respond with $\text{sk} \leftarrow \text{Derive}(\text{msk}, \text{id})$.
 - A single “challenge” query on two identities $\text{id}_0^*, \text{id}_1^*$. Choose a random bit $b \in \{0, 1\}$ and compute $\text{sk}^* \leftarrow \text{Derive}(\text{msk}, \text{id}_b^*)$; There is no reply.
- After the challenge query is made, \mathcal{A} can additionally make arbitrary “ciphertext” queries on ciphertexts c . Respond with $\text{Dec}(\text{sk}_{\text{id}_b^*}, c)$.
- Finally, \mathcal{A} outputs a guess b' for b .

Definition 2. A traitor tracing scheme Π is indistinguishably private (IND-P) if, for all PPT adversaries \mathcal{A} , there exists a negligible function negl such that $\Pr[b' = b] \leq 1/2 + \text{negl}(\lambda)$.

Note that our definition allows for id_b^* to also be asked during identity queries. We can also consider some variations on the above notion:

- We can limit N to be at most some value c (which may depend on λ). We say such a scheme is c -bounded collusion indistinguishably private (c -IND-P).
- Alternatively, we can imagine giving \mathcal{A} the master secret key msk in the clear at the beginning of the experiment. In this case, we note that identity queries are redundant, as the adversary can now run `Derive` for himself. This setting captures the case where the master secret key may unintentionally be leaked, or alternatively where the key distributor is initially honest but later becomes corrupted. In this case, we say the scheme is *leaked master indistinguishably private* (LM-IND-P).

Simulation-based notions. We can also define *simulation*-based notions of privacy for traitor tracing, which require that the responses to ciphertext queries can be simulated without knowing the identity. More precisely, we consider two experiments, called the “Real world” and the “Ideal world”. The “Real world” is identical to the experiment above, except that the challenge query consists of a single identity id^* and $\text{sk}^* \leftarrow \text{Derive}(\text{msk}, \text{id}^*)$. In the “Ideal world”, the experiment is the same, except that sk^* is never computed, and ciphertext queries on ciphertext c are answered by a simulator $S(\text{msk}, c, r)$ that does not know id^* . Here, r is some randomness that is chosen at the beginning of the experiment, and used for every ciphertext query. Let W_R, W_I be the probabilities \mathcal{A} outputs 1 in the Real/Ideal world, respectively.

Definition 3. A traitor tracing scheme Π is simulation private (SIM-P) if there exists a simulator S such that, for all PPT adversaries \mathcal{A} , there exists a negligible function negl such that $|\Pr[W_R = 1] - \Pr[W_I = 1]| \leq \text{negl}(\lambda)$.

We can define c -SIM-P and LM-SIM-P analogously. Note that we always give S the master secret key msk ; this is in some sense necessary, since S somehow must be able to decrypt ciphertexts in order to simulate.

It is easy to show that the indistinguishability and corresponding simulation notions are equivalent, by having S simply compute sk for an arbitrary id , and answer all decryption queries with sk . Thus we have that:

Lemma 1. A traitor tracing scheme Π is IND-P (respectively c -IND-P/LM-IND-P) if and only if it is SIM-P (resp. c -SIM-P/LM-SIM-P).

Impossibility of Black Box Traceable Private Traitor Tracing. Here, we show that black box traceable private traitor tracing is impossible:

Theorem 1. If Π is black box 1-traceable⁹, then it is not even 0-IND-P.

⁹ Recall that c -traceable means the adversary gets $\leq c$ secret keys.

Proof. Let \mathcal{A} be the following adversary for privacy: on input pk, msk , it outputs two random (distinct) identities $\text{id}_0^*, \text{id}_1^*$. It also chooses two random distinct messages m_0, m_1 . Let $\epsilon = 1/2$. \mathcal{A} runs $A \leftarrow \text{Trace}^{\text{Dec}}(\text{pk}, m_0, m_1, 1^1, 1^{1/\epsilon})$, where Dec uses \mathcal{A} 's decryption oracle to decrypt ciphertexts c to get m , and then outputs 1 if and only if $m = m_1$. If A contains exactly one of id_b^* , output b ; otherwise output a random bit. By the correctness of Π , with probability $1 - \text{negl}$, D is a “good” decoder and so $\text{GoodDec}_\epsilon(\mathcal{A}, \lambda)$ happens. As D only depends on the single secret key $\text{sk}_{\text{id}_b^*}$, 1-traceability means $A = \{\text{id}_b^*\}$ with probability $1 - \text{negl}$. As such, \mathcal{A} will output b with probability $1 - \text{negl}$, breaking 0-IND-P. \square

3.3 Consistent Traitor Tracing

In this section, we give our new definition of consistency for traitor tracing. Let \mathcal{A} be an adversary, and consider the following experiment:

- \mathcal{A} gets input 1^λ , and produces a number N .
- Run $(\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N)$, and send pk to \mathcal{A} . \mathcal{A} can now make two kinds of queries, in any order:
 - At most N “identity” queries on identities id ; respond with $\text{sk} \leftarrow \text{Derive}(\text{msk}, \text{id})$. Let T be the set of id queried.
 - A single “challenge” query on two identities $\text{id}_0^*, \text{id}_1^*$. Compute $\text{sk}_b^* \leftarrow \text{Derive}(\text{msk}, \text{id}_b^*)$ for $b = 0, 1$; there is no reply.
 Throughout the experiment, we require $\text{id}_0^*, \text{id}_1^* \notin T$, or else we immediately abort and set the output of the experiment to **Lose**.
- After the challenge query is made, \mathcal{A} can additionally make arbitrary “ciphertext” queries on ciphertexts c . For such queries, compute $m_b = \text{Dec}(\text{sk}_{\text{id}_b^*}, c)$ for $b = 0, 1$. If $m_0 = m_1$, respond with m_0 . Otherwise, immediately abort and set the output of the experiment to **Win**.
- At the end of the experiment, if no abort happened, output **Lose**.

Definition 4. A traitor tracing scheme Π is weakly consistent (W-CONSIS) if, for all PPT adversaries \mathcal{A} , there exists a negligible function negl such that $\Pr[\text{Win}] \leq \text{negl}(\lambda)$.

Note that we can consider numerous variants of consistency:

- We can bound N by c , in which case Π is c -bounded collusion weakly consistent (c -W-CONSIS).
- We can give \mathcal{A} the master secret key msk , in which case we say that Π is leaked master weakly consistent (LM-W-CONSIS). Note that in this case, identity queries are redundant.
- Instead of having separate challenge and ciphertext queries, we can simply ask the adversary to produce a ciphertext that results in different decryption outcomes among the secret keys controlled by the adversary. In other words, the adversary cannot find a differing input amongst his secret keys, even though he has them in the clear. In this case, we say that Π is strongly consistent (S-CONSIS). Bounded collusion strong consistency and leaked master strong consistency are defined similarly.

Relation to Privacy. Here, we discuss the relationship between privacy and consistency. We observe that consistency actually implies privacy: if the adversary cannot find a ciphertext on which two secret keys decrypt differently, then anything it can learn by querying the secret key must be independent of the identifying information. This is formalized by the following:

Theorem 5. *If Π is W-CONSIS (respectively LM-W-CONSIS or c-W-CONSIS), then it is also IND-P (resp. LM-IND-P, c-IND-P).*

Proof. We prove the case W-CONSIS \Rightarrow IND-P, the other cases being proved similarly. Let \mathcal{A} be an adversary for IND-P; we use \mathcal{A} to construct an adversary \mathcal{A}' for W-CONSIS. \mathcal{A}' runs \mathcal{A} , answering all queries by forwarding all queries to its own challenger. Notice that \mathcal{A}' perfectly simulates the view of \mathcal{A} , up until \mathcal{A} makes a ciphertext query where the secret keys sk_0^*, sk_1^* result in different outcomes. But in this case, \mathcal{A}' will forward the query and win.

Suppose that \mathcal{A}' does not win. Conditioned on this case, \mathcal{A} learns nothing about b since its queries would be answered identically with both sk_0^*, sk_1^* . As such, the probability \mathcal{A} wins is exactly $1/2$. Overall, if \mathcal{A}' wins with probability ϵ , \mathcal{A} wins with probability at most $1/2 + \epsilon$. By the assumed W-CONSIS security, ϵ must be negligible. We conclude that IND-P holds. \square

As an immediate corollary of Theorem 5, we have:

Corollary 1. *If Π is black box 1-traceable, then it is not even 0-W-CONSIS.*

4 Functional Encryption and Black Box Privacy

In this section, we discuss functional encryption and introduce a new notion of privacy called black box function privacy. A functional encryption scheme is tuple $\Pi_{FE} = (\text{Gen}, \text{Enc}, \text{Derive}, \text{Dec})$ of PPT algorithms:

$$\begin{aligned} (\text{pk}, \text{msk}) &\leftarrow \text{Gen}(1^\lambda, N) & c &\leftarrow \text{Enc}(\text{pk}, m) \\ \text{sk}_f &\leftarrow \text{Derive}(\text{msk}, f) & o &\leftarrow \text{Dec}(\text{sk}_f, c) . \end{aligned}$$

Above, λ is the security parameter, N an upper bound on the number of users, pk the public key, msk the master secret key, f a function, sk_f a function-specific secret key, m a message, c a ciphertext, and o an output. We require that Dec recovers $f(m)$: there exists a negligible negl such that for all $\lambda > 0, N, f, m$:

$$\Pr \left[\text{Dec}(\text{sk}_f, f, c) = f(m) : \begin{array}{l} (\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{sk}_f \leftarrow \text{Derive}(\text{msk}, f) \\ c \leftarrow \text{Enc}(\text{pk}, m) \end{array} \right] \geq 1 - \text{negl}(\lambda) .$$

If the probability above is identically 1, we say the scheme is *perfectly* correct.

Ciphertext Indistinguishability. We now recall the “usual” definition of security for functional encryption [BSW10, O’N10], which we will call *ciphertext indistinguishability*. Consider the following experiment on an adversary \mathcal{A} :

- \mathcal{A} gets input 1^λ , and produces a number N .
- Run $(\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N)$ and send pk to \mathcal{A} . \mathcal{A} can now make two kinds of queries, in any order:
 - Up to N “function” queries on functions f ; Return $\text{sk}_f \leftarrow \text{Derive}(\text{msk}, f)$.
 - A single “challenge” query on a pair of messages m_0, m_1 . Choose a random bit $b \in \{0, 1\}$ and reply with $\text{Enc}(\text{pk}, m_b)$.
 The only restriction on m_0, m_1 and the various f is that $f(m_0) = f(m_1)$.
- Finally, \mathcal{A} outputs a guess b' for b .

Definition 5. Π_{FE} is adaptively ciphertext indistinguishable (IND-C) if for all \mathcal{A} , there exists a negligible negl such that $\Pr[b' = b] \leq 1/2 + \text{negl}(\lambda)$.

We also consider a c -bounded collusion (c -IND-C) version where $N \leq c = c(\lambda)$.

Known Results. Indistinguishability obfuscation (plus one-way functions) implies functional encryption with adaptive ciphertext indistinguishability [Wat14, ABSV15]. Public key encryption implies c -bounded collusion functional encryption for any polynomial c , where the parameters of the system grows polynomially in c [GVW12]. These schemes are perfectly correct.

4.1 Black Box Function Privacy

We now consider the privacy of f . Function privacy has been considered before (e.g. [BS15, AAB+13]), however it has always previously tried to keep f private even given sk_f . Note that for *public key* functional encryption, we can always construct from sk_f a circuit $C_f(x) = \text{Dec}(\text{sk}_f, \text{Enc}(\text{pk}, x))$ which computes f . Here, we consider a *stronger* notion of privacy, but in a weaker threat model. We do not care about hiding f from the user who holds sk_f ; instead, we want to hide f from other *remote* users mounting a chosen ciphertext attack against the holder of sk_f . Now, an adversary can query $f(x)$ by querying sk_f on $\text{Enc}(\text{pk}, x)$. By Barak et al.’s impossibility result [BGI+01], such queries may reveal less than the actual code C_f , allowing for stronger privacy in the black box model.

Our Definition. Our formalization black box function privacy uses the Real/Ideal paradigm. Let f^* be a function and consider the following “Real” experiment $\text{BB-FP-Exp}_{\text{Real}}^{f^*}(\mathcal{A}, \lambda)$ between an adversary \mathcal{A} and a challenger:

- \mathcal{A} gets input 1^λ , and produces a number N .
- Run $(\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N)$ and send pk and msk to \mathcal{A} . Run $\text{sk}^* \leftarrow \text{Derive}(\text{msk}, f^*)$; sk^* is kept secret.
- \mathcal{A} can now make an arbitrary number of “ciphertext” queries, where it produces a ciphertext c . In response, it gets $\text{Dec}(\text{sk}^*, c)$.
- Finally, \mathcal{A} outputs a bit b , which is the output of the experiment.

Also consider the “Ideal” experiment $\text{BB-FP-Exp}_{Ideal}^{f^*}(S, \lambda)$ for a “simulator” S . The Ideal experiment is identical to the Real experiment, except “ciphertext” queries are replaced with “function” queries, where S sends x and receives $f^*(x)$, and sk^* is never generated. Note that by giving \mathcal{A}, S the master secret key, \mathcal{A} and S can always compute function secret keys on its own.

Definition 6. A functional encryption scheme Π_{FE} is black box function private (BB-FP) if, for every PPT \mathcal{A} , there exists a PPT simulator S and a negligible negl such that, for every function f^* ,

$$\left| \Pr[1 \leftarrow \text{BB-FP-Exp}_{Real}^{f^*}(\mathcal{A}, \lambda)] - \Pr[1 \leftarrow \text{BB-FP-Exp}_{Ideal}^{f^*}(S, \lambda)] \right| < \text{negl}(\lambda).$$

4.2 Upgrading to Black Box Function Privacy

In the full version [Zha21], we show the following:

Theorem 6. Assuming the existence of NIZKs and a functional encryption scheme that is both perfectly correct and IND-C (resp. c-IND-C), then there exists a functional encryption scheme that is perfectly correct, IND-C (resp. c-IND-C) and BB-FP.

The proof idea is simple: the simulator runs the adversary, decrypting any ciphertext query it receives to learn the input x , which it then sends as a function query. To decrypt, the simulator obtains the secret key for the identity function by using the master secret key. The potential problem is that the adversary may try to devise a ciphertext which decrypts inconsistently under the identity secret key vs the secret key for the function f^* . To overcome this problem, we include a zero-knowledge proof of well-formedness, which combined with (perfect) correctness guarantees that decryption will be consistent.

5 Constructing Private Traitor Tracing

In this section, we give our private traitor tracing scheme with whitebox tracing. Our construction will consist of two pieces: we first build a *function-embedded* traitor tracing scheme, which allows for embedding functions, rather than identities. We then embed un-obfuscatable functions into the scheme. The result allows for tracing, while maintaining privacy.

5.1 Function-Embedded Traitor Tracing (FETT)

Here, we introduce and construct *function-embedded traitor tracing* (FETT). The rough idea of a FETT is that user secret keys have *functions* embedded in them, rather than just data. The tracing algorithm will take as an additional input x , and will output the embedded function f evaluated on x . While our ultimate goal is to construct a white box tracing scheme, this part of our construction will actually leverage prior techniques and will therefore be black box.

We will require a notion of black box function privacy analogous to functional encryption, where having oracle access to the decryption function of a secret key with f embedded is “no better than” having black box access to f itself. Black box function privacy implies that we *cannot* simply use identity-embedded traitor tracing (e.g. [NWZ16]) where we set the identity to be some (perhaps obfuscated) code of f . Indeed, in such a solution tracing would recover the code of f . Looking forward to our private tracing construction, black box function privacy allows us to embed an un-obfusctatable function (UOF), and use the inability to learn the UOF under black box queries to argue privacy.

In order to formalize our notion of tracing, we actually break tracing into two steps that we call **FindTags** and **Eval**. The first step, **FindTags**, extracts from a decoder a list of “tags” that were used in generating the user secret keys. The tracing guarantee insists that the recovered tags correspond to users controlled by the adversary. We note that these tags are independent of the function f . We then have a second step, **Eval**, which takes as input a tag and an input x , and computes $f(x)$, where f is the function embedded in the secret key associated with the given tag.

In the case of colluding users, these tags allow for disambiguating between the functions controlled by the adversary, both in the construction and also in the definition. We now give the full definition: a FETT is a tuple Π_{FETT} where:

$$\begin{aligned} (\text{pk}, \text{msk}) &\leftarrow \text{Gen}(1^\lambda, N) & \text{sk} &\leftarrow \text{Derive}(\text{msk}, f, \tau) \\ c &\leftarrow \text{Enc}(\text{pk}, m) & A &\leftarrow \text{FindTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^N, 1^{1/\epsilon}) \\ m &\leftarrow \text{Dec}(\text{sk}, c) & o &\leftarrow \text{Eval}^{\text{D}}(\text{pk}, m_0, m_1, 1^N, 1^{1/\epsilon}, \tau, x) . \end{aligned}$$

Above, $\lambda, \text{pk}, \text{msk}, \text{sk}, m, c, \text{D}, m_0, m_1, N, \epsilon$ are the same as in plain traitor tracing. τ is a tag from set Γ and x is an input. Correctness is similar to standard traitor tracing: there exists a negligible function negl such that for all $\lambda > 0, N, f, m$:

$$\Pr \left[\text{Dec}(\text{sk}, c) = m : \begin{array}{l} (\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \tau \leftarrow \Gamma \\ \text{sk} \leftarrow \text{Derive}(\text{msk}, f, \tau) \\ c \leftarrow \text{Enc}(\text{pk}, m) \end{array} \right] \geq 1 - \text{negl}(\lambda) .$$

Consider the following experiment on adversary \mathcal{A} and parameter $\epsilon = \epsilon(\lambda)$:

- \mathcal{A} gets input 1^λ , and produces a number N .
- Run $(\text{pk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, N)$. Send pk to \mathcal{A} .
- \mathcal{A} then makes an arbitrary number of queries on functions f_i . For each query, respond with $\text{sk} \leftarrow \text{Derive}(\text{msk}, \tau_i)$, where $\tau_i \leftarrow \Gamma$ is chosen randomly. Let T be the set of τ_i generated.
- \mathcal{A} produces a decoder D , two messages m_0, m_1 , and an input x^* .
- Run $A \leftarrow \text{FindTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^{|T|}, 1^{1/\epsilon})$.
- Additionally, let $y_i = f_i(x^*)$ and $y'_i \leftarrow \text{Eval}^{\text{D}}(\text{pk}, m_0, m_1, 1^{|T|}, 1^{1/\epsilon}, \tau_i, x^*)$ for each i such that $\tau_i \in A$.

We define the following events. $\text{BadTr}_\epsilon(\mathcal{A}, \lambda), \text{GoodDec}_\epsilon(\mathcal{A}, \lambda), \text{GoodTr}_\epsilon(\mathcal{A}, \lambda)$ are as before: BadTr means $A \not\subseteq T$, GoodDec means $\Pr[\text{D}(\text{Enc}(\text{pk}, m_b)) = b] \geq 1/2 + \epsilon(\lambda)$, and GoodTr means $|A| > 0$. Finally, $\text{Incorrect}_\epsilon(\mathcal{A}, \lambda)$ means $y_i \neq y'_i$ for some i such that $\tau_i \in A$.

Definition 7. Π_{FETT} is traceable if, for all PPT \mathcal{A} and inverse-poly ϵ , there exists a negligible negl such that $\Pr[\text{BadTr}_\epsilon(\mathcal{A}, \lambda)] \leq \text{negl}(\lambda)$, $\Pr[\text{GoodTr}_\epsilon(\mathcal{A}, \lambda)] \geq \Pr[\text{GoodDec}_\epsilon(\mathcal{A}, \lambda)] - \text{negl}(\lambda)$, and $\Pr[\text{Incorrect}_\epsilon(\mathcal{A}, \lambda)] \leq \text{negl}(\lambda)$.

Note that the first two inequalities correspond to the standard tracing guarantee, with the τ_i playing the role of identities. The final inequality corresponds to the requirement that the function computed by `Eval` matches the function embedded in the secret key, at least on inputs computable by the adversary.

We also need a notion of black box function privacy for traitor tracing. The definition is syntactically *identical* to that of black box function privacy, and so we omit the formal definition. However, we note that the function f plays a different role in functional encryption vs traitor tracing: in functional encryption, the secret key for a function f recovers $f(m)$. In function-embedded traitor tracing, the secret key for a function f recovers m entirely. Yet in both cases it is possible to query f on arbitrary inputs, either through encryption or through tracing. Black box function privacy in both cases means, essentially, that you cannot do better than black box queries.

5.2 From BB Private FE to FETTs

We now show how to use functional encryption with black box function privacy to build function-embedded traitor tracing. Our construction is an adaptation of a construction from Nishimaki et al. [NWZ16]. Specifically, they give a variant of their main construction which achieves short ciphertexts, despite having secret keys with large embedded identities. In this work, we do not focus on the sizes of parameters, but the general structure of their construction will be useful for us.

Construction 1. Let $\Pi_{\text{FE}} = (\text{Gen}_{\text{FE}}, \text{Enc}_{\text{FE}}, \text{Derive}_{\text{FE}}, \text{Dec}_{\text{FE}})$ be a functional encryption scheme. We will assume without loss of generality that all functions considered output a single bit; we can convert any long-output function into a single-bit function by providing an additional input which selects the desired output bit. Let $\Pi_{\text{FETT}} = (\text{Gen}_{\text{FETT}}, \text{Enc}_{\text{FETT}}, \text{Derive}_{\text{FETT}}, \text{Dec}_{\text{FETT}}, \text{FindTags}, \text{Eval})$ be the following traitor tracing scheme:

- $\text{Gen}_{\text{FETT}} = \text{Gen}_{\text{FE}}$.
- $\text{Enc}_{\text{FETT}}(\text{pk}, m) = \text{Enc}_{\text{FE}}(\text{pk}, (0, \perp, m))$.
- $\text{Derive}_{\text{FETT}}(\text{pk}, g, \tau) = \text{Derive}_{\text{FE}}(\text{pk}, f_{g,\tau})$ where $\tau \leftarrow [2^\lambda]$ and

$$f_{g,\tau}(z, x, m) = \begin{cases} m & \text{if } \tau > z \\ \perp & \text{if } (\tau < z) \vee (\tau = z \wedge x = \perp) \\ m & \text{if } \tau = z \wedge x \neq \perp \wedge g(x) = 1 \\ \perp & \text{if } \tau = z \wedge x \neq \perp \wedge g(x) = 0 \end{cases} .$$

Note that z is allowed to range from 0 to 2^λ , and x comes from the domain of g , or x can be the special symbol \perp .

- $\text{Dec}_{\text{FETT}}(\text{sk}, c) = \text{Dec}_{\text{FE}}(\text{sk}, c)$.

Before describing tracing in detail, we first give an intuition for the above construction. Under normal operation, $z = 0$ and $x = \perp$, meaning that $f_{\tau,g}$ outputs m . Therefore, the scheme is correct. We also note that black box function privacy of Π_{FEET} follows immediately from the black box function privacy of Π_{FE} .

For tracing, **FindTags** runs the oracle jump-finding algorithm of Nishimaki et al. [NWZ16], essentially performing a binary search to find a list of τ ; each τ value has the property that there is a significant “jump” in decryption probability between $z = \tau$ and $z = \tau - 1$. In this part, $x = \perp$, and so the function g is never used. Then **Eval** will set x to be the desired input and $z = \tau$, and see which side of the jump the decryption probability is closer to. Functional encryption security implies that the decoder cannot tell whether the ciphertext contains $(\tau - g(x), \perp, m)$ or (τ, x, m) (since both cases decrypt identically), and as a result the decryption probability in **Eval** will indicate the value of $g(x)$.

$\text{FindTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^{|T|}, 1^{1/\epsilon})$. Our **FindTags** algorithm is essentially the tracing algorithm of Nishimaki et al. [NWZ16]. Consider $x = \perp$; then $f_{g,\tau}(z, \perp, m)$ is m if $\tau > z$ and is \perp if $\tau \leq z$. As such, we can write $f_{\tau}(z, m) = f_{g,\tau}(z, \perp, m)$, which is independent of g . Thus, by setting $\text{Enc}(\text{pk}, (z, m)) = \text{Enc}_{\text{FE}}(\text{pk}, (z, x = \perp, m))$, we obtain a private linear broadcast encryption scheme (PLBE) [BSW06] with the tags τ as the indices. Nishimaki et al. show how to trace such a scheme in the case that the tag space is super-polynomial; due to lack of space, we defer the details to the Full Version [Zha21], and paraphrase their main result. We will call their tracing algorithm **FindTags**. Let $p(\tau) = \Pr[\text{D}(\text{Enc}_{\text{FE}}(\tau, \perp, m_b))] = b : b \leftarrow \{0, 1\}$.

Theorem 7 ([NWZ16]). *There exists a PPT $\text{FindTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^{|T|}, 1^{1/\epsilon})$ and inverse polynomial $\delta = \delta(\epsilon, |T|, \lambda)$ such that, for all PPT adversaries \mathcal{A} and all inverse-polynomials ϵ , there exists a negligible function negl such that*

- $\Pr[\text{BadTr}_{\epsilon}(\mathcal{A}, \lambda)] \leq \text{negl}(\lambda)$,
- $\Pr[\text{GoodTr}_{\epsilon}(\mathcal{A}, \lambda)] \geq \Pr[\text{GoodDec}_{\epsilon}(\mathcal{A}, \lambda)] - \text{negl}(\lambda)$,
- *Except with probability $\text{negl}(\lambda)$, $|p(\tau) - p(\tau - 1)| \geq 4\delta$ for all $\tau \in A$.*

$\text{Eval}^{\text{D}}(\text{aux}, \tau, x)$. Define $q(\tau, x) = \Pr[\text{D}(\text{Enc}_{\text{FE}}(\tau, x, m_b)) = b : b \leftarrow \{0, 1\}]$. **Eval** will compute estimates $\hat{p}(\tau), \hat{p}(\tau - 1), \hat{q}(\tau, x)$ of $p(\tau), p(\tau - 1), q(\tau, x)$ by making $O(\lambda/\delta^2)$ queries to **D** on ciphertexts with plaintext $(\tau, \perp, m_b), (\tau - 1, \perp, m_b), (\tau, x, m_b)$, respectively. The result is:

Lemma 2. *Except with negligible probability in λ , $|\hat{p}(\tau) - p(\tau)|, |\hat{p}(\tau - 1) - p(\tau - 1)|$, and $|\hat{q}(\tau, x) - q(\tau, x)|$ are $< \delta$.*

Now, notice that the τ_i are all distinct with overwhelming probability, and that $f_{g,\tau}(\tau, x, m) = f_{g,\tau}(\tau - g(x), \perp, m)$, while $f_{g,\tau'}(\tau, x, m)$ for $\tau' \neq \tau$ is independent of g . Therefore, by ciphertext indistinguishability, encryptions of (τ_i, x, m) are indistinguishable from encryptions of $(\tau_i - g_i(x), \perp, m)$, for any x that can be produced by the adversary. As a result, $|q(\tau_i, x) - p(\tau_i - f(x))| < \text{negl}(\lambda)$. Thus, $|\hat{q}(\tau_i, x) - \hat{p}(\tau_i - f(x))| < 2\delta$, while $|\hat{q}(\tau_i, x) - \hat{p}(\tau_i - (\neg f(x)))| > 2\delta$.

Therefore, `Eval` outputs b such that $\hat{q}(\tau_i, x)$ is closer to $\hat{p}(\tau_i - b)$ than to $\hat{p}(\tau_i - (-b))$. By the above, we have that $\Pr[\text{Incorrect}_\epsilon(\mathcal{A}, \lambda)] \leq \text{negl}(\lambda)$.

Putting everything together, we have the following theorem:

Theorem 8. *If Π_{FE} is IND-C, then Π_{FETT} is traceable. If Π_{FE} is black box function private, then so is Π_{FETT} .*

5.3 Un-Obfuscatable Functions (UOFs)

We will make use of un-obfuscatable functions (UOFs). These were originally constructed by Barak et al. [BGI+01]. Here, we define a variant which differs in a few key ways from that of Barak et al.:

- We allow for embedding messages into the program.
- We only require that the embedded message can be reconstructed from the program code, whereas Barak et al. reconstruct the entire function.
- For technical reasons, we must allow the obfuscated code to be randomized and have differing inputs from the original code. We instead require, essentially, that as long as differing inputs are hard to find, it is possible to learn the embedded message.

More formally, an un-obfuscatable function $\Pi_{\text{UOF}} = (\text{Sample}, \text{Extract}, \text{Diff})$ is a tuple of PPT algorithms:

$$f \leftarrow \text{Sample}(1^\lambda, m; r) \quad m \leftarrow \text{Extract}(f') \quad x \leftarrow \text{Diff}(f', r, m, 1^{1/\epsilon}) .$$

Above, m is a message, r the random coins for `Sample`, f' a probabilistic circuit, and ϵ a parameter. We require two security properties. First is *black box unlearnability*, where we require that it is impossible to learn anything about m with just black box access to f . Concretely, consider the following experiment on an adversary \mathcal{A} :

- \mathcal{A} , on input the security parameter 1^λ , produces two messages m_0, m_1 . Choose a random bit b and compute $f \leftarrow \text{Sample}(1^\lambda, m_b)$.
- \mathcal{A} now can now make arbitrary queries to f .
- Finally, \mathcal{A} produces a guess b' for b .

We say that Π_{UOF} is *black box unlearnable* if, for any PPT adversary \mathcal{A} , there exists a negligible negl such that $\Pr[b' = b] < 1/2 + \text{negl}(\lambda)$.

The second security requirement is *reverse engineerability*, where we require that m can be learned given any code f' computing m . For technical reasons, we need to allow f' to be randomized and also have some differing inputs from f . But if m cannot be learned, then it must be possible to actually compute such differing inputs. In more detail, consider the following experiment on adversary \mathcal{A} and parameter ϵ :

- \mathcal{A} , on input the security parameter 1^λ , produces a message m . Reply with $f \leftarrow \text{Sample}(1^\lambda, m; r)$ for random coins r .
- \mathcal{A} then outputs f' . Run $m' \leftarrow \text{Extract}(f')$ and $x \leftarrow \text{Diff}(f', r, m, 1^{1/\epsilon})$.

Let $\text{Goodfunc}_\delta(\mathcal{A}, \lambda)$ be the event that $\Pr[f'(x) \neq f(x)] < \delta(\lambda)$ where the probability in \Pr is over the random coins of f' . Let $\text{BadExtr}_\epsilon(\mathcal{A}, \lambda)$ be the event that f' satisfies $\Pr[\text{Extract}(f') \neq m] \geq \epsilon(\lambda)$, where the probability in \Pr is over the random coins on Extract . We say that Π_{UOF} is *reverse engineerable* if, for every PPT \mathcal{A} and inverse polynomial ϵ , there exists a inverse polynomial δ and negligible negl such that $\Pr[\text{BadExtr}_\epsilon(\mathcal{A}, \lambda) \wedge \text{Goodfunc}_\delta(\mathcal{A}, \lambda)] < \text{negl}(\lambda)$. Note that in Barak et al.'s notion, f' and f are required to compute identical functions, Goodfunc is always guaranteed to happen and there is no need to consider Diff .

Definition 8. *An un-obfuscatable function Π_{UOF} is secure if it is black box unlearnable and reverse engineerable.*

Barak et al.'s construction. In the Full Version [Zha21], we recall Barak et al.'s construction of an unobfuscatable function, and show that it satisfies our definition; in particular, we show an algorithm Extract which extracts a differing input from any program where extraction fails. The result is the following:

Theorem 9. *Assuming one-way functions, there exists a secure un-obfuscatable function.*

5.4 Our Private Traitor Tracing Scheme

We now turn to our private tracing scheme.

Construction 2. *Let $\Pi_{\text{FETT}} = (\text{Gen}, \text{Enc}, \text{Derive}_{\text{FETT}}, \text{Dec}, \text{FindTags}, \text{Eval})$ be a FETT and $\Pi_{\text{UOF}} = (\text{Sample}, \text{Extract}, \text{Diff})$ a UOF. Define the new traitor tracing scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Derive}, \text{Dec}, \text{Trace})$ where*

- $\text{Derive}(\text{msk}, \text{id})$: *Return $\text{sk} \leftarrow \text{Derive}_{\text{FETT}}(\text{msk}, \text{Sample}(1^\lambda, \text{id}))$.*
- $\text{Trace}(\text{pk}, \text{D}, m_0, m_1, 1^N, 1^{1/\epsilon})$: *Produce $A \leftarrow \text{FindTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^N, 1^{1/\epsilon})$. Then, for each $\tau_i \in A$, run $\text{id}_i \leftarrow \text{Extract}(P_i)$ where P_i is the (randomized) program $x \mapsto \text{Eval}^{\text{D}}(\text{pk}, m_0, m_1, 1^N, 1^{1/\epsilon}, \tau_i, x)$. Output $\{\text{id}_i : \tau_i \in A\}$.*

The correctness of Π is immediate. We now discuss security:

Theorem 10. *If Π_{FETT} is traceable and black box function private and if Π_{UOF} is secure, then Π is traceable and leaked master indistinguishably private.*

Proof. First, we consider privacy. By black box function privacy, the view of a privacy adversary can be simulated by making black box queries to the functions f . But by the black box unlearnability of Π_{UOF} , such queries reveal nothing about the identities. Thus, privacy follows.

We now consider tracing. Let \mathcal{A} be a tracing adversary for Π . We construct a new adversary \mathcal{A}' for Π_{FETT} . \mathcal{A}' simulates \mathcal{A} . For each secret key query on identity id_i , \mathcal{A}' runs $f_i \leftarrow \text{Sample}(1^\lambda, \text{id}_i; r_i)$ with fresh randomness r_i and makes a secret key query on f_i , which it forwards to \mathcal{A} . \mathcal{A}' thus perfectly simulates the view of \mathcal{A} . Now, once \mathcal{A} produces a decoder D , \mathcal{A}' outputs D . Additionally, \mathcal{A}' runs the (public) algorithm FindTags on D to collect a set A' of tags τ_i . Next,

\mathcal{A}' matches the τ_i to the r_i, id_i , and constructs the programs P_i as in `Trace`. It then runs $x_i \leftarrow \text{Diff}(P_i, r_i, \text{id}_i, 1^{1/\epsilon'})$ for an ϵ' to be specified later. It outputs D and sets x^* to be a random choice amongst the x_i .

Consider running `Trace` on D , and let A be the set of τ_i recovered by `FindTags` when run as a part of `Trace`. By the tracing security of Π_{FETT} , we know that, except with negligible probability, `FindTags` correctly outputs a subset of the tags τ_i generated during the adversary's identity queries. It remains to show that `Trace` correctly recovers the corresponding id_i . Let p be the probability that A is indeed a subset of the correct τ_i , but for some i^* , the recovered identity $\text{id}'_{i^*} := \text{Extract}(P_{i^*})$ is such that $\text{id}'_{i^*} \neq \text{id}_{i^*}$. We must show that p is negligible. Suppose toward contradiction p is non-negligible, and let ϵ' be a polynomial which lower bounds $(p/N)^2$ infinitely often; note that p is determined entirely by \mathcal{A}, ϵ , meaning we can set ϵ' freely.

Suppose we choose i^* at random from A ; then with probability at least p/N , we will have that $\text{id}'_{i^*} \neq \text{id}_{i^*}$. Note also that, once \mathcal{A} outputs the decoder D , A and the A' generated by \mathcal{A}' are two samples from identical distributions. As a consequence, with probability at least $(p/N)^2 \geq \epsilon'$, the x^* produced by \mathcal{A}' is equal to x_{i^*} and $\text{id}'_{i^*} \neq \text{id}_{i^*}$. Moreover, the P_{i^*} constructed by \mathcal{A}' is identical to the P_{i^*} constructed inside `Trace`. Therefore, by the reverse engineerability of Π_{UOF} , we must have that x^* is a differing input: there exists a δ such that $\Pr[P_{i^*}(x^*) \neq f(x^*)] \geq \delta$. This implies that, in the FETT experiment, $y_{i^*} := f_{i^*}(x^*)$ and $y'_{i^*} := P_{i^*}(x^*)$ differ with non-negligible probability. In other words, $\Pr[\text{Incorrect}_\epsilon(\mathcal{A}', \lambda)]$ is non-negligible, contradicting the security of Π_{FETT} . \square

6 Toward Consistent Traitor Tracing

Here, we give our construction of consistent traitor tracing. We first recall some additional definitions.

Fully Homomorphic Encryption. Fully homomorphic encryption (FHE) can be built from circularly-secure variants of `LWE` [Gen09, GSW13], or from sub-exponentially secure `iO` and “lossy” encryption [CLTV15]. An FHE scheme is a tuple $\Pi_{\text{FHE}} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$ where $(\text{Gen}, \text{Enc}, \text{Dec})$ form a public key encryption scheme. Additionally, $\text{Eval}(c, f)$ takes as input a ciphertext and a circuit f , with the property that, for every polynomial poly , there exists a negligible function such that, for every $\lambda > 0, x$ and f such that $|f| \leq \text{poly}(\lambda)$:

$$\Pr \left[\text{Dec}(\text{sk}, \text{Eval}(c, f)) = f(x) : \begin{matrix} (\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda) \\ c \leftarrow \text{Enc}(\text{pk}, x) \end{matrix} \right] \geq 1 - \text{negl}(\lambda) .$$

Compute and Compare Obfuscation. Compute and compare obfuscation can be built from `LWE` [GKW17, WZ17], or seen as a special case of `iO`. It consists of a PPT algorithm $\text{CCObf}(C, \beta)$ which takes as input a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^\lambda$, and a value $\beta \in \{0, 1\}^\lambda$. It outputs a circuit $C' : \{0, 1\}^n \rightarrow \{0, 1\}$ such that:

- Correctness: $C'(x) = 1$ if $C(x) = \beta$, and $C'(x) = 0$ otherwise.
- Security: There exists an algorithm Sim with the following guarantee. For any C , let β be chosen uniformly from $\{0, 1\}^\lambda$. Then for any PPT \mathcal{A} , there exists a negligible negl such that

$$\left| \Pr[\mathcal{A}(\text{CCObf}(C, \beta)) = 1] - \Pr[\mathcal{A}(\text{Sim}(1^{|C|}, 1^n, 1^\lambda)) = 1] \right| < \text{negl}(\lambda) .$$

In other words, if β is uniform, then C' reveals nothing about C, β .

6.1 The Construction

Construction 3. Fix a constant c . Let $\Pi_{\text{FHE}} = (\text{Gen}_{\text{FHE}}, \text{Enc}_{\text{FHE}}, \text{Dec}_{\text{FHE}}, \text{Eval}_{\text{FHE}})$ be a fully homomorphic encryption scheme, $\Pi_{\text{FE}} = (\text{Gen}_{\text{FE}}, \text{Enc}_{\text{FE}}, \text{Derive}_{\text{FE}}, \text{Dec}_{\text{FE}})$ a functional encryption scheme, $\Pi_{\text{Sig}} = (\text{Gen}_{\text{Sig}}, \text{Sign}, \text{Ver})$ a signature scheme, F a PRF, and CCObf a compute and compare obfuscation scheme. Define the traitor tracing scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Derive}, \text{Dec}, \text{Trace})$ as follows:

- $\text{Gen}(1^\lambda, N)$: run $(\text{pk}_{\text{FE}}, \text{msk}_{\text{FE}}) \leftarrow \text{Gen}_{\text{FE}}(1^\lambda, N)$. Also run $(\text{pk}_{\text{Sig}}, \text{sk}_{\text{Sig}}) \leftarrow \text{Gen}_{\text{Sig}}(1^\lambda)$. For each $i \in [N]$, choose a random key $k^i \leftarrow \{0, 1\}^\lambda$. Then, for every $S \subseteq [N]$ such that $|S| \leq c$, let $\sigma^S \leftarrow \text{Sign}(\text{sk}_{\text{Sig}}, S)$, run $(\text{pk}^S, \text{sk}^S) \leftarrow \text{Gen}_{\text{FHE}}(1^\lambda)$, $c^S \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}^S, \sigma^S)$, $\beta^S \leftarrow \bigoplus_{i \in S} \text{F}(k^i, S)$ and finally $P^S \leftarrow \text{CCObf}(\text{Dec}_{\text{FHE}}(\text{sk}^S, \cdot), \beta^S)$. Output $\text{pk} = (\text{pk}_{\text{FE}}, (\text{pk}^S, c^S, P^S)_S)$ and $\text{msk} = (\text{msk}_{\text{FE}}, (k^i)_{i \in [N]}, \text{ctr} = 0)$.
- $\text{Enc}(\text{pk}, m) = \text{Enc}_{\text{FE}}(\text{pk}_{\text{FE}}, (\perp, \perp, \perp, \perp, m))$
- $\text{Derive}(\text{msk}, \text{id})$: Run $\text{sk} \leftarrow \text{Derive}_{\text{FETT}}(\text{msk}_{\text{FE}}, h_{\text{id}, i})$ where $i = \text{ctr}$ and

$$h_{\text{id}, i}(S, \sigma, j, x, m) = \begin{cases} m & \text{if } \text{Ver}(\text{pk}_{\text{Sig}}, S, \sigma) = 0 \vee i \in S \setminus \{j\} \\ g_{\text{id}, i}(S, j, x, m) & \text{if } \text{Ver}(\text{pk}_{\text{Sig}}, S, \sigma) = 1 \wedge i \notin S \setminus \{j\} \end{cases} ,$$

$$g_{\text{id}, i}(S, j, x, m) = \begin{cases} \perp & \text{if } i \neq j \vee x = \perp \\ m & \text{if } i = j \wedge f_{\text{id}, i}(S, x) = 1 \quad , \text{ and} \\ \perp & \text{if } i = j \wedge f_{\text{id}, i}(S, x) = 0 \end{cases}$$

$$f_{\text{id}, i}(S, x = (b, u)) = \begin{cases} \text{F}(k^i, S)_u & \text{if } b = 0 \\ \text{id}_u & \text{if } b = 1 \end{cases} .$$

Also increment ctr within msk . Here, $\text{F}(k^i, S)_u$ is the u th bit of $\text{F}(k^i, S)$.

We note that our construction requires a stateful Gen , which keeps a counter. This is to ensure that the tags used for different users are unique. An alternative, similar to what was done in [GKW19], would be to have Derive take the tag as an explicit input, and assume some external mechanism to ensure distinct tags.

Before formally giving the tracing algorithm and proving security, we discuss the intuition behind the above construction. Consider encryptions of plaintexts

$(S, \sigma^S, \perp, \perp, m)$. Since $g_{\text{id},i}(S, \perp, \perp, m) = \perp$, such ciphertexts can be decrypted by users with indices $i \in S$, but cannot be decrypted by $i \notin S$. Let $p(S)$ be the probability a decoder decrypts such ciphertexts.

For any good decoder, we must have $p([N])$ be large. FE security implies $p(Q)$ is close to $p([N])$, where Q is the set of indices the adversary controls. Moreover, FE security implies that $p(\emptyset)$ is close to 0. A straightforward argument implies that there must therefore exist a set $S^* \subseteq Q$ such that $p(S^*)$ is noticeably larger than $p(S^* \setminus \{i\})$ for all $i \in S^*$. Supposing we could sign arbitrary sets, we can recover S^* by estimating the various $p(S)$ values.

On the other hand, the ability to sign sets S also allows for easily finding differing inputs, which would break consistency. Instead, we can use the encryptions of signatures to homomorphically compute $p(S)$. Unfortunately, we cannot directly compare different $p(S)$, since the signatures for different S , and therefore the $p(S)$, are isolated in different FHE instances. However, given an (encrypted) signature on S , we can (homomorphically) estimate $p(S \setminus \{i\})$ for any $i \in S$ by testing the decoder on ciphertexts encrypting $(S, \sigma^S, i, \perp, m)$. Indeed, these ciphertexts can only be decrypted by users in $S \setminus \{i\}$, and functional encryption security implies that encryptions of $(S, \sigma^S, i, \perp, m)$ are indistinguishable from encryptions of $(S \setminus \{i\}, \sigma^{S \setminus \{i\}}, \perp, \perp, m)$.

For the set S^* (which at this point is still FHE encrypted), we can then run the decoder (again, homomorphically) on encryptions of $(S^*, \sigma^{S^*}, i, x, m)$. Depending on the value of $f_{\text{id},i}(S^*, x)$, the decryption probability will either be roughly $p(S^*)$ or $p(S^* \setminus \{i\})$; since the definition of S^* means the two probabilities are noticeably different, this allows us to learn $f_{\text{id},i}(S^*, x)$. From here, we can compute $F(k^i, S^*)$ by setting $b = 0$, and hence β^{S^*} .

Up until this point, we cannot actually tell which set is S^* , since all results are computed homomorphically and therefore still hidden under FHE encryptions. We perform the above procedure for each set S , as there are only polynomial many. We then apply the program P^S to the resulting ciphertext, which will output 1 in the case $S = S^*$. This allows us to actually determine S^* .

The next step is to determine the identities for users in S^* . We show that any accepting input to P^{S^*} actually allows us to decrypt ciphertexts encrypted under pk^{S^*} ; in particular we can find σ^{S^*} in the clear. This then allows us to evaluate $f_{\text{id},i}(S^*, x)$ on arbitrary inputs x in the clear. Using such queries we can easily compute the various id by setting $b = 1$.

It remains to justify consistency, which follows from the fact that β^S is pseudorandom as long as S contains honest users. By applying compute and compare security, we have that σ^S remains hidden for such sets. Since the adversary cannot obtain a signature on any S containing honest users, any ciphertext he devises will be decrypted correctly by all honest users; in particular, all honest users answer identically.

6.2 Tracing

We now give the algorithm $\text{Trace}(\text{pk}, \text{D}, m_0, m_1, 1^c, 1^{1/\epsilon})$. Define

$$\begin{aligned} p(S) &= \Pr[\text{D}(\text{Enc}_{\text{FE}}(\text{pk}_{\text{FE}}, (S, \sigma^S, \perp, \perp, m_b))) = b : b \leftarrow \{0, 1\}] \\ p(S, i) &= \Pr[\text{D}(\text{Enc}_{\text{FE}}(\text{pk}_{\text{FE}}, (S, \sigma^S, i, \perp, m_b))) = b : b \leftarrow \{0, 1\}] \\ q(S, i, x) &= \Pr[\text{D}(\text{Enc}_{\text{FE}}(\text{pk}_{\text{FE}}, (S, \sigma^S, i, x, m_b))) = b : b \leftarrow \{0, 1\}] . \end{aligned}$$

Let $\delta = \epsilon/(10c + 2)$. For any S , given σ^S we can compute an estimates $\tilde{p}(S), \tilde{p}(S, i), \tilde{q}(S, i, x)$ such that $|\tilde{p}(S) - p(S)|, |\tilde{p}(S, i) - p(S, i)|, |\tilde{q}(S, i, x) - p(S, i, x)| < \delta$, except with negligible probability. Each quantity is computed by making $O(\lambda/\delta^2)$ queries to D . We define several subroutines:

- $\text{ConfirmTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, S, \sigma)$: This algorithm plays an analogous role as FindTags from Sect. 5, except that instead of discovering a set of accused users, it simply confirms whether the input set S should be accused. In this sense, ConfirmTags works in the black box confirmation model of [BF99]. The algorithm is also somewhat different that of Sect. 5, owing to the different tracing structure in this construction. Compute estimate $\tilde{p}(S)$, and for each $j \in S$, compute estimates $\tilde{p}(S, j)$. If there exists a $j \in S$ such that $|\tilde{p}(S, j) - \tilde{p}(S)| < 4\delta$, abort and output \perp . Otherwise, output $\text{aux} = (\tilde{p}(S), (\tilde{p}(S, j))_{j \in S})$.
- $\text{Eval}^{\text{D}}(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, \text{aux}, S, \sigma, j, x)$: This algorithm is analogous to Eval from Sect. 5. Compute estimate $\tilde{q}(S, j, x)$, and output 1 such that $\tilde{q}(S, j, x)$ is closer to $\tilde{p}(S)$ than it is to $\tilde{p}(S, j)$; otherwise output 0.
- $\text{Dec}^*(\text{pk}^S, c^S, P^S, C, d)$: here, C is a circuit, with the property that $C(\sigma^S) = \beta^S$, meaning $P^S(\text{Eval}_{\text{FHE}}(c^S, C)) = 1$; d is a ciphertext encrypting a bit b . Dec^* will output b . Dec^* works as follows. It homomorphically computes d' , an encryption of $b \cdot \sigma^S$, from d, c^S . Then it will run and output $P^S(\text{Eval}_{\text{FHE}}(d', C))$.

With these subroutines in hand, Trace works as follows. Let $C^S(\sigma)$ be the function which runs ConfirmTags to recover aux or \perp . If aux is recovered, then for each $i \in S$, it runs $\text{Eval}^{\text{D}}(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, \text{aux}, S, \sigma, i, x)$ on the various $x = (0, u)$ to compute strings $\beta^{S, i} = \text{F}(k^i, S)$. Finally, it outputs $\bigoplus_{i \in S} \beta^{S, i}$. The circuit C^S is ostensibly randomized, but we will hard-code the randomness to get a deterministic circuit.

For each set S , we will say that tracing succeeds if $P^S(\text{Eval}(c^S, C^S)) = 1$, which is equivalent to requiring $C^S(\sigma^S) = \beta^S$. For each S , Trace runs $\text{Dec}^*(\text{pk}^S, c^S, P^S, C^S, d = c^S)$. Let S be some set such that Dec^* outputs a signature σ . Then Trace runs $\text{ConfirmTags}^{\text{D}}(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, S, \sigma)$ to recover aux , and runs $\text{Eval}^{\text{D}}(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, \text{aux}, S, \sigma, i, x)$ on various $x = (1, u)$ to compute the bits of id^i for $i \in S$.

6.3 Security

Theorem 11. *If Π_{FE} is ciphertext indistinguishable and black box function private, and $\Pi_{\text{FHE}}, \Pi_{\text{Sig}}, \text{F}, \text{CCObf}$ are secure, then Π in Construction 3 is c -traceable and leaked master weakly consistent.*

Proof. We first prove leaked master weak consistency. Let \mathcal{A} be an adversary for consistency. By the black box function privacy of Π_{FE} , there exists a simulator Sim that only makes queries to the functions $h_{id,i}$ of the various honest users, and can still find a differing input with non-negligible probability. In particular, it must with non-negligible probability find a query (S^*, σ, z, x, m) to some $h_{id,i}$ such that σ is a valid signature on S^* and $i \in S^*$. Let q be the index of the first query where this happens. For all prior queries, $h_{id,i}$ outputs m . Therefore, all prior queries can be simulated without knowing a signature on any S that contains honest users, and also without knowing k^i for any honest user i .

For every honest user i , we can therefore replace each evaluation of $\beta^{S,i}$ with random. This change will be undetectable before query q , by the PRF security of F . But this means, by compute and compare security, that $P^{S,i}$ can be simulated without knowing $\text{sk}^{S,i}$, which again will be undetectable before query q . We finally rely on the security of $\text{pk}^{S,i}$ to conclude that the entire view of the adversary up until query q can be simulated just knowing σ^S , where S ranges over all subsets containing only adversarial users.

The result is that the view of the adversary up until query q can be simulated by making signing queries on S containing only adversarial users, but then query q produces a signature on an S^* containing at least one honest user, which must therefore be different than any of the queries S . Thus, such an algorithm can forge signatures, a contradiction to the security of pk_{Sig} .

We now prove c -traceability. Consider an attacker \mathcal{A} which makes up to c queries. Let Q be the set of $i \in [N]$ corresponding to the adversary's queries. Let D be the output of \mathcal{A} , and suppose GoodDec_ϵ happens. We note that for any honest user $i \notin Q$, by functional encryption security $p(S)$ and $p(S \setminus \{i\})$ will be negligibly close, except with negligibly-small probability. As such, honest users will never be accused. We now prove that some user will be accused.

Claim. Except with negligible probability, $p(Q) > 1/2 + \epsilon - \delta$ and $p(\emptyset) < 1/2 + \delta$

Proof. Under all the adversary's keys, $(\perp, \perp, \perp, \perp, m)$ and $(Q, \sigma^Q, \perp, \perp, m)$ decrypt correctly, so encryptions of these values are indistinguishable. $p(Q) > 1/2 + \epsilon - \delta$ follows by the goodness of D . On the other hand, $(\emptyset, \sigma^\emptyset, \perp, \perp, m)$ will always fail to decrypt, so $p(\emptyset) < 1/2 + \delta$ except with negligible probability. \square

Claim. Except with negligible probability, there exists an $S^* \subseteq Q$ such that, for all $i \in S^*$, $p(S^* \setminus \{i\}) \leq p(S^*) - 8\delta$.

Proof. Assume $p(Q) > 1/2 + \epsilon - \delta$ and $p(\emptyset) < 1/2 + \delta$. Suppose toward contradiction that, for each set $S \subseteq Q$, there exists an i_S such that $p(S \setminus \{i_S\}) > p(S) - 8\delta$. Then setting $S_0 = Q$ and $S_j = S_{j-1} \setminus \{i_{S_{j-1}}\}$, we get that $p(S_j) > p(S_{j-1}) - 8\delta$ and $S_{|Q|} = \emptyset$. But this means that $p(\emptyset) > p(Q) - 8\delta|Q|$, a contradiction. \square

Claim. Except with negligible probability, $|p(S^* \setminus \{j\}) - p(S^*, j)| < \delta$ for any $i \in S^*$.

Proof. For any $i \in S^*$ and any secret key under the adversary's control, $(S^* \setminus i, \sigma^{S^* \setminus i}, \perp, \perp, m)$ and $(S^*, \sigma^{S^*}, i, \perp, m)$ decrypt identically. Therefore, their encryptions are indistinguishable. \square

By the above claims, $p(S^*, i) < p(S^*) - 8\delta$ for each $i \in S^*$, except with negligible probability. But then $\tilde{p}(S^*, i) < \tilde{p}(S^*) - 6\delta$ except with negligible probability. When we homomorphically run $\text{FindTags}_0^D(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, S^*, \sigma^*)$, no abort will happen and the result will be (an encryption of) aux .

Claim. For any $i \in S^*$ and x , if $f_{\text{id},i}(S^*, x) = 0$ then $|q(S^*, i, x) - p(S^*, i)| < \delta$ except with negligible probability. If $f_{\text{id},i}(S^*, x) = 1$, then $|q(S^*, i, x) - p(S^*)| < \delta$.

Proof. If $f_{\text{id},i}(S^*, x) = 0$, then the secret key for user i rejects encryptions of $(S^*, \sigma^{S^*}, i, x, m)$, while all other users in S^* decrypt and users outside S^* reject. This is the same functionality as $(S^*, \sigma^{S^*}, i, \perp, m)$. On the other hand, if $f_{\text{id},i}(S^*, x) = 1$, then the secret key for user i correctly decrypts $(S^*, \sigma^{S^*}, i, x, m)$, corresponding to the same functionality as $(S^*, \sigma^{S^*}, \perp, \perp, m)$. The claim follows by functional encryption security. \square

Claim. For any $i \in S^*$ and any input x , except with negligible probability $\text{Eval}_0^D(\text{pk}, m_0, m_1, 1^c, 1^{1/\epsilon}, \text{aux}, S^*, \sigma^{S^*}, i, x)$ outputs $f_{\text{id},i}(S^*, x)$.

Proof. If $f_{\text{id},i}(S^*, x) = 0$, then $|\tilde{q}(S^*, i, x) - \tilde{p}(S^*, i)| < 3\delta$. But since $|\tilde{p}(S^*) - \tilde{p}(S^*, i)| > 6\delta$, we must have $|\tilde{q}(S^*, i, x) - \tilde{p}(S^*)| > 3\delta$. As such, $\tilde{q}(S^*, i, x)$ is closer to $\tilde{p}(S^*, i)$ than $\tilde{p}(S^*)$, and Eval_0 therefore outputs 0 on input x . Analogously, Eval_0 outputs 1 on inputs x such that $f_{\text{id},i}(S^*, x) = 1$. \square

Therefore, the circuit $C^{S^*}(\sigma^{S^*})$ will correctly evaluate $\beta^{S^*, i} = F(k^i, S^*)$, and therefore correctly output β^{S^*} with overwhelming probability. \square

References

- [AAB+13] Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: Functional encryption and property preserving encryption: new definitions and positive results. Cryptology ePrint Archive, Report 2013/744 (2013). <http://eprint.iacr.org/2013/744>
- [ABSV15] Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_32
- [ADVW13] Agrawal, S., Dodis, Y., Vaikuntanathan, V., Wichs, D.: On continual leakage of discrete log representations. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 401–420. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_21
- [BF99] Boneh, D., Franklin, M.K.: An efficient public key traitor tracing scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_22

- [BGI+01] Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
- [BN08] Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 2008, pp. 501–510. ACM Press, October 2008
- [BP13] Bitansky, N., Paneth, O.: On the impossibility of approximate obfuscation and applications to resettable cryptography. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 241–250. ACM Press, June 2013
- [BS15] Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_12
- [BSW06] Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_34
- [BSW10] Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. Cryptology ePrint Archive, Report 2010/543 (2010). <http://eprint.iacr.org/2010/543>
- [BZ14] Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_27
- [CFN94] Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_25
- [CGZ20] Cohen, R., Garay, J.A., Zikas, V.: Broadcast-optimal two-round MPC. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 828–858. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_28
- [CHN+16] Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC, pp. 1115–1127. ACM Press, June 2016
- [CLTV15] Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_19
- [DF03] Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 100–115. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_8
- [Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 169–178. ACM Press, May/June 2009
- [GKRW18] Goyal, R., Koppula, V., Russell, A., Waters, B.: Risky traitor tracing and new differential privacy negative results. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 467–497. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_16

- [GKW17] Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: Umans, C. (ed.) 58th FOCS, pp. 612–621. IEEE Computer Society Press, October 2017
- [GKW18] Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) 50th ACM STOC, pp. 660–670. ACM Press, June 2018
- [GKW19] Goyal, R., Koppula, V., Waters, B.: New approaches to traitor tracing with embedded identities. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 149–179. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_6
- [GNPT13] Guillot, P., Nimour, A., Phan, D.H., Trinh, V.C.: Optimal public key traitor tracing scheme in non-black box model. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 140–155. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38553-7_8
- [GSW13] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
- [GVW12] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
- [JKL09] Junod, P., Karlov, A., Lenstra, A.K.: Improving the Boneh-Franklin traitor tracing scheme. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 88–104. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_6
- [KD98] Kurosawa, K., Desmedt, Y.: Optimum traitor tracing and asymmetric schemes. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 145–157. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054123>
- [KY03] Kiayias, A., Yung, M.: Breaking and repairing asymmetric public-key traitor tracing. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 32–50. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-44993-5_3
- [NDC+15] Ning, J., Dong, X., Cao, Z., Wei, L., Lin, X.: White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes. *IEEE Trans. Inf. Forensics Secur.* **10**(6), 1274–1288 (2015)
- [NP01] Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45472-1_1
- [NWZ16] Nishimaki, R., Wichs, D., Zhandry, M.: Anonymous traitor tracing: how to embed arbitrary information in a key. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 388–419. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_14
- [O’N10] O’Neill, A.: Definitional issues in functional encryption. *Cryptology ePrint Archive, Report 2010/556* (2010). <http://eprint.iacr.org/2010/556>
- [Pfi96] Pfitzmann, B.: Trials of traced traitors. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 49–64. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61996-8_31

- [TNS06] Tonien, D., Safavi-Naini, R.: An efficient single-key pirates tracing scheme using cover-free families. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 82–97. Springer, Heidelberg (2006). https://doi.org/10.1007/11767480_6
- [Wat14] Waters, B.: A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588 (2014). <http://eprint.iacr.org/2014/588>
- [WZ17] Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: Umans, C. (ed.) 58th FOCS, pp. 600–611. IEEE Computer Society Press, October 2017
- [Zha20] Zhandry, M.: New techniques for traitor tracing: size $N^{1/3}$ and more from pairings. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 652–682. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_22
- [Zha21] Zhandry, M.: White box traitor tracing (full version) (2021)



Does Fiat-Shamir Require a Cryptographic Hash Function?

Yilei Chen¹(✉), Alex Lombardi², Fermi Ma^{3,4}, and Willy Quach⁵

¹ Tsinghua University, Beijing, China
chenyilei@mail.tsinghua.edu.cn

² MIT, Cambridge, USA
alexjl@mit.edu

³ Princeton University, Princeton, USA
fermima@alum.mit.edu

⁴ NTT Research, Sunnyvale, USA

⁵ Northeastern University, Boston, USA
quach.w@husky.neu.edu

Abstract. The Fiat-Shamir transform is a general method for reducing interaction in public-coin protocols by replacing the random verifier messages with deterministic hashes of the protocol transcript. The soundness of this transformation is usually *heuristic* and lacks a formal security proof. Instead, to argue security, one can rely on the *random oracle methodology*, which informally states that whenever a random oracle soundly instantiates Fiat-Shamir, a hash function that is “sufficiently unstructured” (such as fixed-length SHA-2) should suffice. Finally, for some special interactive protocols, it is known how to (1) isolate a concrete security property of a hash function that suffices to instantiate Fiat-Shamir and (2) build a hash function satisfying this property under a cryptographic assumption such as Learning with Errors.

In this work, we abandon this methodology and ask whether Fiat-Shamir truly requires a cryptographic hash function. Perhaps surprisingly, we show that in two of its most common applications—building signature schemes as well as (general-purpose) non-interactive zero-knowledge arguments—there are sound Fiat-Shamir instantiations using extremely simple and non-cryptographic hash functions such as $\text{mod-}p$ or bit decomposition. In some cases, we make idealized assumptions (i.e., we invoke the generic group model), while in others, we prove soundness in the plain model.

On the negative side, we also identify important cases in which a cryptographic hash function is provably necessary to instantiate Fiat-Shamir. We hope this work leads to an improved understanding of the precise role of the hash function in the Fiat-Shamir transformation.

The full version of this paper is available [16].

A. Lombardi—Research supported in part by an NDSEG fellowship. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, and by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

© International Association for Cryptologic Research 2021
T. Malkin and C. Peikert (Eds.): CRYPTO 2021, LNCS 12828, pp. 334–363, 2021.
https://doi.org/10.1007/978-3-030-84259-8_12

1 Introduction

The Fiat-Shamir transform is a general-purpose method for converting public-coin interactive protocols into *non-interactive* protocols with the same functionality. As a prototypical example, let Π denote a 3-message (public-coin) argument system with transcripts of the form (α, β, γ) . Then, given any *hash function* h , the Fiat-Shamir transform of Π using h , denoted $\Pi_{\text{FS},h}$, is a one-message argument system in which the prover sends an entire transcript $(\alpha, \beta = h(\alpha), \gamma)$ in one shot.

The Fiat-Shamir transform was introduced by [27] to remove interaction from a 3-message identification scheme, but it was later realized¹ that the transformation is extremely general: it can plausibly be applied to *any* constant-round public-coin interactive argument system (and more). Due to its generality and its *practical efficiency* (it removes interaction with very low computational overhead), the transformation has been a cornerstone of both theoretical and practical cryptography for over 30 years. Some of its applications include the construction of efficient signature schemes [27, 50, 52], non-interactive zero-knowledge arguments (NIZKs) [1, 11, 12, 49], and succinct non-interactive arguments (SNARGs) [2–7, 36, 43, 56].

However, the vast majority of applications of the Fiat-Shamir transform are only *heuristically sound*. That is, the resulting non-interactive protocols do not have proofs of soundness based on the computational intractability of a well-studied mathematical problem [32]. Nonetheless, the protocols appear to be sound in practice, so it has been a long-standing goal of theoretical cryptography to *justify* the soundness of the transformation.

So far, there have been two main approaches for justifying soundness of Fiat-Shamir.

- **The Random Oracle Model** [1]: In this design methodology, a Fiat-Shamir hash function is first modeled as a random function \mathcal{O} to which all parties (honest and dishonest) have public query access. Security is “argued” by showing that the protocol $\Pi_{\text{FS},\mathcal{O}}$ is sound “in the random oracle model” (i.e., against query-bounded adversaries). In reality, the hash function h is instantiated by an “unstructured” hash function (such as SHA-2 on bounded-length inputs), where the implicit expectation is that “Fiat-Shamir for Π ” is not an application that can distinguish h from a random oracle.
- **Correlation Intractability**: In a recent line of work [9, 11, 12, 33, 34, 37, 49], a different methodology was developed for provably instantiating Fiat-Shamir in the standard model:
 - Identify a special class \mathcal{C} of protocols and a cryptographic security property \mathcal{P} of a hash function family \mathcal{H} such that if \mathcal{H} satisfies \mathcal{P} , then \mathcal{H} soundly instantiates Fiat-Shamir for every $\Pi \in \mathcal{C}$. In all cases so far, \mathcal{P} has been a restricted form of correlation intractability [13].
 - Construct a hash function family satisfying \mathcal{P} under reasonable (hopefully standard) cryptographic assumptions.

¹ See discussion in [1].

The first of these approaches attempts to justify the use of Fiat-Shamir in high generality, while the second provides full security proofs for carefully chosen protocols and hash functions.

Why Cryptographic Hash Functions? In both approaches above, it is essential that the hash function h possesses a form of *cryptographic hardness*. In the random oracle methodology, it is heuristically assumed that h is indistinguishable from a truly random function (at least in any meaningful way), while in the standard model, results so far have relied on correlation-intractable hash families [13, 47] whose security can be based on standard cryptographic assumptions [9, 11, 49].

All of these results support the intuition that the Fiat-Shamir hash family \mathcal{H} provides a form of cryptographic hardness that ensures the soundness of $\Pi_{\text{FS}, \mathcal{H}}$. In this work, we ask whether this intuition is accurate.

Is it possible to instantiate the Fiat-Shamir heuristic with a non-cryptographic hash function?

We note that this question requires formalizing what it means to be a “non-cryptographic” (rather than cryptographic) hash function; we partially address this issue later, but this remains somewhat up to interpretation.

A related question concerns the *design* of Fiat-Shamir hash functions. What should they look like? Again, prior works give us some possible answers:

- As originally proposed in [27], a Fiat-Shamir hash function could be instantiated using a pseudorandom function family [31] (they give DES as an example instantiation).
- As proposed in the random oracle methodology [1], the following design advice is given. “When instantiating a random oracle by a concrete function h , care must be taken first to ensure that it is adequately conservative in its design so as not to succumb to cryptanalytic attack, and second to ensure that h exposes no relevant ‘structure’ attributable to its being designed from some lower-level primitive.” In other words, the hash function should be *unstructured* and *complex* enough to be indistinguishable from a random function.
- In the provably secure instantiations of [11, 49], the hash function families are based on flavors of *fully homomorphic encryption*, which can be instantiated from lattice assumptions [10, 29].
- In a recent work of [9], a (modified) *trapdoor hash function* [25] is used, which has instantiations based on the DDH/LWE/QR/DCR assumptions.

A common theme is that all of the candidate Fiat-Shamir hash functions above are *complex*. Indeed, they have to be complex enough to realize the described security properties. In contrast, we ask:

Is it possible to instantiate Fiat-Shamir with a simple hash function?

As an example, can we hope to have a *linear* Fiat-Shamir hash function $h(x) = Ax + b$?

We note that for various contrived protocols Π , the answer is “yes” for uninteresting reasons. For example, given any constant-round, public-coin interactive protocol Π , there is a protocol $\tilde{\Pi}$ that replaces all prover messages α_i with random-oracle commitments $\mathcal{O}(\alpha_i)$ and requires the prover to open these commitments in the last round. For this protocol $\tilde{\Pi}$, even the identity function can be used to instantiate Fiat-Shamir in the random oracle model, since we have in effect *already* applied a random-oracle Fiat-Shamir transformation when converting Π to $\tilde{\Pi}$.

To avoid these trivialities, we phrase our goal more specifically: for various *naturally occurring* protocols (or classes of naturally occurring protocols), determine if simple/non-cryptographic hash functions may suffice for Fiat-Shamir, and give principled justification for this possibility or impossibility.

1.1 Our Contributions

We begin the systematic study of instantiating Fiat-Shamir with simple and non-cryptographic hash functions. In particular, we focus on two common and important use cases of Fiat-Shamir:

1. Round-compressing 3-message identification schemes [27, 40, 52], and
2. Round-compressing 3-message honest-verifier zero knowledge argument systems to obtain NIZK arguments for NP [1, 9, 11, 12, 17, 21, 49].

For these two use cases, we identify some common 3-message protocols to which Fiat-Shamir is applied:

- Schnorr’s identification scheme [52].
- The Chaum-Pedersen interactive proof system for the Diffie-Hellman language [15].
- Lyubashevsky’s lattice-based identification scheme [40].
- More generally, Σ -protocols [23], which are typically repeated in parallel to obtain negligible soundness error.

In this work, we consider whether existing protocols from above can be round-compressed using a simple/non-cryptographic hash function. We are able to show both negative results and (perhaps surprisingly) *positive* results on this front.

Before stating our results more formally, we discuss (1) the specific problems we want to solve and (2) what constitutes a solution to the problem.

Our Methodology. There are two major issues to resolve in order to define our problem:

- (i) What does it mean for a hash function to be *cryptographic*?
- (ii) How do we give evidence for the soundness (or lack thereof) of our round-compressed protocols?

We first partially address question (i). One appealing intuitive definition of a cryptographic hash function is as follows:

Definition 1 (Cryptographic Hash Function, definition attempt). *A hash function h (or hash function family \mathcal{H}) is cryptographic if there is a game \mathcal{G} between a challenger and adversary (who is given h or $h \leftarrow \mathcal{H}$) with a statistical-computational gap; that is, the maximum probability that a computationally bounded adversary can win \mathcal{G} is noticeably smaller than the maximum probability that an unbounded adversary can win \mathcal{G} .*

Unfortunately, this definition has major issues. In particular, under a literal interpretation of the definition, if $\text{NP} \not\subseteq \text{BPP}$, then *every* hash function is “cryptographic”: just define the game \mathcal{G} that ignores the hash family \mathcal{H} and gives the adversary an instance of a hard NP problem to solve.

More specific to our application, the soundness of $\Pi_{\text{FS},\mathcal{H}}$ is precisely a game with a computational-statistical gap so long as an accepting proof exists but is computationally hard to find. Therefore, no matter how “simple” or “non-cryptographic” \mathcal{H} appears to be, as long as it can compile Fiat-Shamir for some protocol, it is necessarily “cryptographic” under this definition.

Indeed, an important philosophical point in this work is that the “computational hardness” within the soundness property of $\Pi_{\text{FS},\mathcal{H}}$ can derive from two different places: the **hash family \mathcal{H}** and the **interactive protocol Π** .

For our purposes, we appeal to the following intuitive (non-technical) definition of a cryptographic hash function:

Definition 2 (Cryptographic Hash Function, intuition-level). *Informally, a hash function h (or hash function family \mathcal{H}) is cryptographic if there is a game \mathcal{G} between a challenger and adversary with a statistical-computational gap that does not derive from some separate hard problem.*

Given this partial answer to question (i), we now describe how we handle (ii):

How We Give Positive Results. In order to obtain a positive result, we accomplish (at least) one of three things:

- We show that any hash function h (or hash family \mathcal{H}) satisfying an *information-theoretic property* (e.g., pairwise-independence) suffices to instantiate $\Pi_{\text{FS},\mathcal{H}}$ soundly. We believe that in spirit, this says that Fiat-Shamir for Π does not require a cryptographic hash function (Definition 2), as a purely information theoretic property should be insufficient to establish computational hardness.
- We show that a *single fixed hash function h* (rather than a distribution on hash functions) is enough to soundly instantiate $\Pi_{\text{FS},h}$. More specifically, we show “average-case soundness”, i.e., soundness on a random NO-instance. This is at least enough to strongly distinguish our Fiat-Shamir instantiations from random-oracle hash functions as well as correlation-intractable hash functions, which crucially rely on the randomness of the hash function to derive computational hardness.

- We instantiate $\Pi_{\text{FS},h}$ with an *extremely simple* hash function h , such as a linear function modulo a prime p or the bit decomposition function $\mathbf{G}^{-1} : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_2^{n \log q}$. This does not directly prove that h is not cryptographic, but it again distinguishes our constructions from prior work, in which the Fiat-Shamir hash functions are comparatively complex (see above). Indeed, they are sufficiently complex to guarantee security properties such as correlation intractability.

While some of our positive results hold in the standard model, others are shown to hold in the (auxiliary-input) generic group model [18, 19, 45, 53, 55]. One might ask why such a result is meaningful—after all, we are replacing one random oracle (the hash function) with another (the generic group labeling). However, the idealized assumptions in our constructions are used quite differently from assuming that a Fiat-Shamir hash function behaves like a random oracle. Indeed, our hash functions are information-theoretic and do not make any calls to the group oracle. As a result, our constructions are examples of *naturally occurring* interactive protocols Π (unlike the contrived example from the introduction) that possess enough hardness to guarantee that $\Pi_{\text{FS},h}$ is sound for *simple* choices of h satisfying only information-theoretic properties.

Additionally, our lower bounds in the GGM suggest candidate schemes over concrete groups (\mathbb{Z}_p^\times and elliptic curve groups) that are plausibly secure. Although interpreting hardness results in the GGM in the standard model requires care [24, 28, 54], we believe that it would be very interesting to understand the real-world security of the resulting (extremely simple!) schemes. We do some preliminary analysis of the concrete schemes—finding non-generic attacks for one of our two GGM-based protocols but not the other—but largely leave these questions open.

How We Give Negative Results. In order to obtain a negative result, we would like to show that for a particular protocol Π , if $\Pi_{\text{FS},\mathcal{H}}$ is sound, then \mathcal{H} necessarily satisfies some concrete cryptographic security property \mathcal{P} . However, as already discussed, such a theorem is not meaningful— \mathcal{P} can just be “the soundness of $\Pi_{\text{FS},\mathcal{H}}$.” In other words, this fails to distinguish between hardness in the hash function family \mathcal{H} from hardness in the protocol Π .

Instead, we switch the order of quantifiers in the theorem statement: we show that there exists a *universal* security property \mathcal{P} such that for any protocol $\Pi \in \mathcal{C}$ in a large class, if a hash function family \mathcal{H} soundly instantiates Fiat-Shamir for Π then \mathcal{H} necessarily satisfies \mathcal{P} . Since \mathcal{P} is independent of the protocol Π , this comes closer to distinguishing \mathcal{H} -hardness from hardness in Π .

However, there is still one issue with the above strategy: NP-completeness also gives a (trivial) universal property \mathcal{P} . To avoid this problem, we prove a *relativizing* result: the same property \mathcal{P} is satisfied by \mathcal{H} even if it instantiates Fiat-Shamir for various protocols $\Pi^{\mathcal{O}(\cdot)}$ that exist relative to an oracle distribution \mathcal{O} . This establishes that the property \mathcal{P} is not “cheating” using NP-completeness. As an example, our negative results will capture the $\{0, 1\}$ -challenge variant of

Schnorr’s identification scheme in the generic group model as well as Blum’s Hamiltonicity protocol [8] instantiated in the random-oracle model.

Finally, we show that hash functions satisfying our property \mathcal{P} imply the existence of one-way functions, the quintessential cryptographic object. This results in a formalization of the statement “one-way functions are necessary to instantiate Fiat-Shamir hash functions for natural protocols.”

As an added bonus, we are also sometimes able to give direct attacks on $\Pi_{\text{FS},\mathcal{H}}$ relative to an oracle (i.e., in the generic group model or the random oracle model). That is, for the idealized protocols, we show unconditional polynomial-query attacks on the non-interactive protocol. This is further evidence that a sound Fiat-Shamir instantiation must sometimes rely on hardness from the hash function family \mathcal{H} , in direct contrast to our positive results.

Our Results. With the above discussion in mind, we are now ready to formally state our results. First, we give several positive results for soundly instantiating Fiat-Shamir with *non-cryptographic* hash functions.

Fiat-Shamir for Lattice-Based Identification Schemes. We first describe our positive results in the standard model, which hold for lattice-based analogues of the Schnorr protocol. In particular, we consider common variants of Lyubashevsky’s identification schemes [38–40], which were designed to obtain efficient signature schemes in the random oracle model via Fiat-Shamir.

We obtain a sound Fiat-Shamir instantiation for the main protocol Π defined in [40]. Our Fiat-Shamir hash function in $\Pi_{\text{FS},h}$ maps \mathbb{Z}_q elements to their bit-decomposition (also known as the \mathbf{G}^{-1} function).

Theorem 1. *Consider Lyubashevsky’s identification scheme over \mathbb{Z}_q in dimension n . Define the hash function $h : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_2^{n \log q}$ as the bit decomposition function*

$$h(v) = \mathbf{G}^{-1}(v).$$

Then, under the Short Integer Solution (SIS) assumption, Fiat-Shamir applied to Lyubashevsky’s scheme using hash function h is sound on random instances.

We note the following interesting details about our result.

- We obtain a meaningful soundness guarantee using a **deterministic hash function**. This stands in contrast to typical Fiat-Shamir instantiations.
- More generally, we prove Theorem 1 for a class of Fiat-Shamir hash functions (including bit-decomposition) satisfying an **information-theoretic property**.
- Most importantly, and uniquely to the lattice setting, we emphasize that soundness is proved in the **standard model**! More specifically, the SIS assumption suffices to argue *average-case* soundness, where soundness requires that a cheating prover cannot convince a verifier to accept on a random instance. We stress that this is the typical soundness notion for the setting of identification/signature schemes and a necessary relaxation for the case of deterministic hash functions.

To contrast this with prior work on Fiat-Shamir in the standard model [9, 11, 12, 34, 49], we note that (1) it was not known how to do Fiat-Shamir for the [40] protocol in the correlation intractability framework, and (2) our Fiat-Shamir compiler uses the bit decomposition function and *not* any form of CI.

Finally, as an extension of Theorem 1, we prove that variants of our protocol Π_{FS} show a surprising connection to Micciancio-Peikert lattice trapdoors [41, 44]. Namely, the prover algorithm in Π_{FS} can be interpreted as a preimage sampling algorithm using a Micciancio-Peikert trapdoor.

Theorem 2 (Informal). *Lattice-based Lyubashevsky signatures using the bit-decomposition Fiat-Shamir hash function are equivalent to lattice-based Hash-and-Sign signatures.*

This highlights a strong connection between two seemingly orthogonal paths to build signatures from lattice-based assumptions: one using lattice trapdoors [14, 30, 44] and the other through the Fiat-Shamir heuristic [38–40]. To the best of our knowledge (see [48]), no such connection was known before. We discuss this connection in more detail in the technical overview.

Schnorr Signatures with a Linear Fiat-Shamir Hash Function. Our next result concerns the Schnorr signature scheme, obtained by applying Fiat-Shamir to Schnorr’s three-message protocol for proving knowledge of a discrete logarithm. We show that for signing *short* messages (i.e. the message space is a sparse subset of \mathbb{Z}_p), this classic application of the Fiat-Shamir paradigm does not seem to require any cryptographic properties from the underlying Fiat-Shamir hash function.

Recall that the Schnorr protocol works over a cryptographic group G of order p , and that the Fiat-Shamir hash function takes as input a group element $g \in G$ along with a message $m \in \mathcal{M}$ to be signed, and outputs an element in \mathbb{Z}_p .

Theorem 3 (Schnorr Signatures with a \mathbb{Z}_p -Linear Hash Function). *Consider the Schnorr signature scheme over a group G of order p , where the message space \mathcal{M} is a sparse subset of \mathbb{Z}_p , i.e. $\mathcal{M} \subset \mathbb{Z}_p$ and $|\mathcal{M}|/\mathbb{Z}_p \leq \text{negl}(\lambda)$. Let ℓ be the maximum bit-length representation of any group element, so that any $g \in G$ can be viewed as $g \in \{0, 1\}^\ell = [2^\ell]$. Define the hash family*

$$h_k(g, m) := g + m + k \pmod{p},$$

where on the right-hand side, g is the integer with binary representation $g \in \{0, 1\}^\ell$.

In the auxiliary-input generic group model [55], the Schnorr signature scheme instantiated using h as the Fiat-Shamir hash function is existentially unforgeable against chosen message attacks (EUF-CMA).

As in the lattice setting, we can actually prove that Fiat-Shamir for Schnorr is sound whenever h (or the family \mathcal{H}) satisfies an information-theoretic property. However, our security proof relies on the GGM and does not seem to carry over

to the standard model. Nonetheless, we view Theorem 3 as another interesting example of a Fiat-Shamir instantiation whose soundness does not rely on any cryptographic property of the hash function. Instead, **strong cryptographic hardness from the group turns out to be sufficient!**

Another takeaway from Theorem 3 is that Schnorr-like signatures can plausibly be obtained by combining a collision-resistant hash function (to implement hash-and-sign) with an information-theoretic Fiat-Shamir hash function (for Schnorr signatures on short messages). While this does not appear significantly different from using a cryptographic Fiat-Shamir hash function *in implementation*, it highlights the fact that cryptographic hashing is required for signatures only to (computationally) avoid *collisions* between long messages, and *not* for ensuring soundness of the Fiat-Shamir compilation.

Aside on Generic Groups. The Generic Group Model [53] models a cryptographic group G as a random injection $G \rightarrow [L]$ for a sufficiently large “label space” L , by providing an oracle \mathcal{O} that computes group products and inverses on (pairs of) labels.² The auxiliary-input GGM [18,55] gives the adversary the additional power to *record* an arbitrary (S -bounded) function of the group’s truth table to use for solving computational problems later.

In the plain GGM, soundness of our variant of Schnorr signatures follows from analysis due to [46]; this work characterized a security property of \mathcal{H} that suffices for (long-message) signatures schemes in the GGM. For our purposes, it turns out that an *information-theoretic* property of h suffices; see Sect. 2 for details. In fact, using the even simpler (keyless) function $h(g, m) = g + m$ is secure in the GGM.

However, since soundness is proved in the GGM, it is reasonable to ask whether the hardness result plausibly translates to concrete groups such as \mathbb{Z}_p^\times or elliptic curve groups. Indeed, it is known that GGM lower bounds sometimes fail to carry over to these groups in cases of interest (see, e.g., [28,54]). In this work, we observe that this issue *also* comes up in the case of Schnorr signatures as analyzed by [46]. In more detail, [46] proves that as long as a hash family \mathcal{H} satisfies two (possibly computational) properties, then Schnorr signatures using \mathcal{H} are secure in the GGM. On the other hand, we find choices of \mathcal{H} that satisfy the premises of [46], but attacks exist over *all concrete groups*. This highlights an important situation where GGM-based analysis spectacularly fails to capture real-world attacks on a scheme.

On the other hand, we further observe that these non-generic attacks can be captured by the auxiliary-input GGM; that is,

² There is an alternative formulation of a Generic Group Model due to Maurer [42], but the *honest parties* in Schnorr’s signature scheme execute non-generic algorithms according to this definition (since Maurer’s GGM does not provide concrete representations of group elements, which are necessary to evaluate the Fiat-Shamir hash function), so a [42]-generic analysis is not applicable.

- Given some (possibly hard-to-compute) short piece of information w about G (but independent of the Schnorr public parameters), Schnorr signatures using \mathcal{H} are insecure, **and**
- Over important concrete groups such as \mathbb{Z}_p^\times or elliptic curve groups, this information w is actually efficiently computable.

For example, the short information could be a solution z to the equation $a^z = \ell$, where $\ell \in [L]$ is a fixed label such that $\ell \equiv -1 \pmod{p}$. To remedy this problem, we prove a lower bound in the aux-input GGM, thus avoiding an important class of “non-generic” attacks for the hash function in Theorem 3 (and more). This proof is the new technical component of Theorem 3.

In fact, we know of no efficient attacks on the scheme from Theorem 3 over the group \mathbb{Z}_p^\times . We find the question of whether this scheme is secure to be interesting, as it would result in a signature scheme that is extremely simple to write down—in fact, key generation, signing, and verifying only require random sampling and arithmetic over \mathbb{Z}_p . We do some preliminary analysis of the scheme in the full version but leave the question largely out of the scope of this paper.

The Chaum-Pedersen Protocol and NIZKs for NP. Next, we consider a minor variant of the interactive proof system due to Chaum and Pedersen [15] for proving membership in the Diffie-Hellman language $\mathcal{L}_{\text{DH}} := \{(g, g^u, g^v, g^{uv})\}_{g \in G, u, v \in \mathbb{Z}_p}$. The protocol was originally introduced to instantiate a (special-purpose) blind signature scheme, but it has since found other applications (e.g., to the Cramer-Shoup cryptosystem [22]). Notably, a recent line of work [20, 21, 35, 51] has shown that a non-interactive, adaptively sound, (single-theorem) zero-knowledge argument for \mathcal{L}_{DH} (along with CDH) suffices to instantiate non-interactive zero-knowledge (NIZK) arguments for all of NP.

We prove in the (auxiliary-input) GGM that a simple, fixed Fiat-Shamir hash function h suffices to compile the modified³ Chaum-Pedersen protocol into an argument for \mathcal{L}_{DH} satisfying an intermediate (i.e., in between selective and adaptive) notion of soundness we call *semi-adaptive* soundness. Here, the prover is given a random g^u , and wins if it convinces the verifier to accept a NO-instance of \mathcal{L}_{DH} of the form (g, g^u, g^y, g^z) .

Theorem 4. *Let Π^{CP} denote the modified Chaum-Pedersen protocol over a group G of order p . Let ℓ be the maximum bit-length representation of any group element, so that any $g \in G$ can be viewed as $g \in \{0, 1\}^\ell = [2^\ell]$. Define the hash function*

$$h(g_1, g_2, g_3, g_4) = g_1 + g_2 + g_3 + g_4 \pmod{p},$$

where on the right-hand side, each g_i is the integer with binary representation $g_i \in \{0, 1\}^\ell$.

In the auxiliary-input generic group model, $(\Pi^{CP})_{\text{FS}, h}$ is a semi-adaptively sound argument system for \mathcal{L}_{DH} .

³ Our modification simply requires the verifier to reject if the third message z is equal to $0 \in \mathbb{Z}_p$.

In the full version, we prove a stronger result: as long as h satisfies an (easily satisfied but complicated to state) information theoretic property, $(\Pi^{\text{CP}})_{\text{FS},h}$ is sound in the aux-input GGM.

By tweaking the hash function to be $h'(\cdot) := h(\cdot) + r$ where r is a common random string, $(\Pi^{\text{CP}})_{\text{FS},h'}$ becomes a (single-theorem) NIZK argument for \mathcal{L}_{DH} with semi-adaptive soundness. It turns out that semi-adaptive soundness suffices to instantiate the hidden bits model of [26], and consequently NIZKs for NP in the standard model [20, 21, 35, 51].

However, we also cryptanalyze this protocol over concrete groups such as \mathbb{Z}_p^\times and elliptic curve groups (see the full version), and unlike the case of Schnorr signatures above, we find non-generic attacks (that fall outside the aux-input GGM) on the scheme. Thus, Theorem 4 should be viewed as a theoretical result that does *not* have direct implications over commonly used groups. This disparity between the GGM and the standard model appears to be quite subtle and deserves further study, as further discussed in our conclusion (Sect. 1.2).

Negative Results. To complement our positive results, we also show that for some protocols, Fiat-Shamir necessarily requires a cryptographic hash function. Our negative results apply to a large class \mathcal{C} of **three-message honest-verifier zero-knowledge (HVZK) arguments** (or proofs), in particular, those obtained by taking parallel repetitions of sigma protocols with polynomial-size challenge space. Two prototypical examples to have in mind are:

- Blum’s Hamiltonicity protocol [8], repeated in parallel to obtain negligible soundness error.
- The one bit challenge variant $\Pi^{\text{bit-Sch}}$ of Schnorr’s identification scheme, again repeated in parallel.

We analyze Fiat-Shamir for these protocols in **both** the standard model and in idealized models (the random-oracle model and the preprocessing GGM, respectively). We give evidence that analogues to Theorem 3, Theorem 4, and Theorem 1 *do not exist* for these protocols. Our two results are as follows.

- **Polynomial-Query Attacks:** First, we show that in idealized models, there will (unconditionally) be a polynomial-query attack on $\Pi_{\text{FS},\mathcal{H}}$, *as long as \mathcal{H} does not depend on the oracle*. In other words, a (poly-query) sound Fiat-Shamir instantiation requires that \mathcal{H} depends on the oracle, which is one way of arguing that \mathcal{H} is cryptographic.

Theorem 5 (Informal). *For $\Pi = \Pi^{\text{bit-Sch}}$ instantiated in the generic group model, if \mathcal{H} is a hash family that does not call the group oracle, then $\Pi_{\text{FS},\mathcal{H}}^t$ is unsound in the GGM.*

For any instantiation of the [8] protocol in the random oracle model, if \mathcal{H} is a hash family that does not depend on the oracle \mathcal{O} , then $\Pi_{\text{FS},\mathcal{H}}$ is unsound.

More generally, for any $\Pi \in \mathcal{C}$ constructed relative to an oracle \mathcal{O} , if \mathcal{H} does not depend on \mathcal{O} , then $\Pi_{\text{FS},\mathcal{H}}$ is unsound.

This is in contrast to Schnorr/Chaum-Pedersen results, in which an oracle-independent hash function suffices for a sound Fiat-Shamir instantiation.

Generalization: What is the class \mathcal{C} ? In full generality (see the full version), the class \mathcal{C} of protocols Π for which we give a polynomial-query attack on $\Pi_{\text{FS}, \mathcal{H}}$ is informally characterized as follows.

- $\Pi := \Pi_{\text{Base}}^t$ is the parallel repetition of a 3-message public-coin HVZK argument system $\Pi_{\text{Base}} = \Pi_{\text{Base}}^{\mathcal{O}(\cdot)}$ (with simulator Sim) relative to an oracle \mathcal{O} .
- The Verifier’s challenge space Σ in Π_{Base} is polynomial-size.
- The underlying language $L \notin \text{BPP}$.
- $(\Pi_{\text{Base}}, \text{Sim})$ is challenge hiding (see the full version).

The last requirement (challenge hiding) is a technical condition that slightly strengthens the standard notion of HVZK.

We emphasize that our result makes no assumptions about the way in which the oracle \mathcal{O} is used in the construction of the interactive protocol Π_{Base} . The most substantial requirement is that Π is the result of *parallel repetition* applied to a protocol with a small (i.e., polynomial) challenge space. This property distinguishes the protocols that we can attack from the protocols for which we find sound Fiat-Shamir instantiations.

- **Conditional Polynomial-time Attacks and Mix-and-Match Resistance:** We describe a concrete security property (which we call “mix-and-match resistance”) such that for any protocol Π in a large class \mathcal{C}' (again including the two example protocols above, *in the standard model*), any hash function (family) \mathcal{H} that instantiates Fiat-Shamir for Π must possess this security property. In other words, we show:

Theorem 6 (Informal). *If \mathcal{H} is not mix-and-match resistant, then for any $\Pi \in \mathcal{C}$, there is a polynomial-time attack on the soundness of $\Pi_{\text{FS}, \mathcal{H}}$.*

At a high level, mix-and-match resistance is a security property asserting the hardness of finding a *combination* of many partial inputs that hashes to a corresponding *combination* of prescribed outputs. We also show that mix-and-match resistant hash functions imply the existence of OWFs. Therefore, Theorem 6 implies that (in the setting above) if $\Pi_{\text{FS}, \mathcal{H}}$ is sound, then \mathcal{H} can be used to build a OWF (obliviously to the protocol Π).

This result also holds in the ROM and the GGM, in the sense that if \mathcal{H} does not depend on the oracle \mathcal{O} and is *not* mix-and-match resistant, then the polynomial-query attack from Theorem 5 can be upgraded to a polynomial-time attack. As discussed above, this further establishes that the “mix-and-match resistance” property of \mathcal{H} is not “borrowing hardness” from the protocol Π , since our analysis applies to protocols whose security is unconditional.

Somewhat orthogonally, one might wonder whether mix-and-match resistant hash functions (as introduced in this work) are known to exist under standard cryptographic assumptions. The works of [11, 49] tell us that the answer is “yes,”

because they give a standard-model instantiation of Fiat-Shamir for a protocol $\Pi \in \mathcal{C}$ under standard assumptions. In the full version, we explore this connection further by showing that correlation-intractable hash functions (as constructed by [11, 49]) suffice to instantiate Fiat-Shamir for (a variant of) the *idealized* Blum protocol.

1.2 Conclusions

One of the main takeaways of this work is that our title question “Does Fiat-Shamir require a cryptographic hash function?” is surprisingly deep and difficult to resolve. We believe that our positive and negative results improve our understanding of the ground truth and point to fascinating new research directions.

Before now, the prevailing intuition was that for any natural protocol (Schnorr, Lyubashevsky, Blum, etc.), sound Fiat-Shamir compilation necessitates a carefully-constructed *cryptographic* hash function. In this methodology, the soundness of Fiat-Shamir has been argued by either (1) treating the hash function as a random oracle or (2) invoking some concrete security property of the function family. That is, the computational hardness of some problem derived from H guarantees the soundness of the protocol.

In this work, we argue soundness of Fiat-Shamir (for certain protocols) by using an *information-theoretic* property of H together with cryptographic hardness from the interactive protocol. Despite the caveats in our results, the conceptual point is clear: it is possible to prove meaningful notions of soundness for a Fiat-Shamir protocol by using security properties of the interactive protocol itself *instead* of security properties of the hash function.

Moreover, the instantiations of our positive results have noticeable qualitative differences from prior approaches to Fiat-Shamir, such as being able to use a *single* hash function h (rather than a family), much simpler hash functions, and ones that contain no associated cryptographic hardness. This contrasts strongly with how we usually think of Fiat-Shamir; essentially all prior work required that the hash function be complex and/or cryptographic.

On the other hand, we also show (and formalize a way to show) that some protocols *do* require a cryptographic Fiat-Shamir hash function. This implies that the ground truth is complicated and hard to characterize, but in our view, worth understanding.

What about Fiat-Shamir in Practice? Since Schnorr signatures are heavily used in practice, one might ask how our positive results over groups relate to the use of Fiat-Shamir over concrete groups. The answer to this question crucially depends on how accurately the generic group model (with preprocessing) reflects the concrete security of these protocols.

While generic group analysis is often considered to be a meaningful reflection of real-world attacks, we discovered multiple non-generic attacks on Fiat-Shamir protocols over groups. Such attacks are therefore not covered by prior generic analyses such as [46].

- In the case of Schnorr signatures over \mathbb{Z}_p^\times , all of the new attacks we found were captured by the *preprocessing* generic group model, and so our new analysis in the preprocessing model rules out all such attacks on many variants of Schnorr signatures. Therefore, we view our positive results for Schnorr as a first step towards finding secure simple variants of Schnorr signatures, such as the candidate given in Construction 11.
- On the other hand, we have already discovered attacks (see the full version) on certain variants of our Chaum-Pedersen protocol over groups such as \mathbb{F}_p^\times , even in settings where we have a valid (preprocessing) generic group analysis.

This results in a bizarre state of affairs in which it is unclear how to interpret generic group analyses for Fiat-Shamir protocols over groups; this deserves future attention and cryptanalytic effort. Nonetheless, we consider the conceptual contributions of these aux-input GGM analyses to be valuable whether they turn out to reflect real-world attacks or not.

Future Work. We believe that our framework can serve as a potential complement to the correlation intractability framework for provable Fiat-Shamir soundness. Towards this end, we broadly ask,

Which interactive protocols allow for “simple” Fiat-Shamir compilers?

To start with, we consider differences between the protocols in our positive and negative results. Heuristically, we note that all protocols in our positive results achieve negligible soundness error using a *single non-separable large challenge*. In contrast, the separability of the challenge in the parallel repetition of a Σ -protocol appears to necessitate using a cryptographic hash function.

In this context, our contributions are a starting point for a more precise understanding of *when* hardness is required from a Fiat-Shamir hash function.

2 Technical Overview

We give an overview of our positive results for lattice-based identification protocols in Sect. 2.1 and our positive results for group-based protocols in Sect. 2.2. We then describe some of our negative results in Sect. 2.3.

2.1 A Non-interactive Lattice-Based Identification Scheme

We describe how we obtain positive results in the lattice setting (Theorem 1). We consider Lyubashevky’s three-message identification protocol [40], which can be seen as a lattice analogue to the Schnorr protocol.

To sample an instance for the protocol, we sample a uniformly random wide matrix \mathbf{A} over \mathbb{Z}_q along with a wide matrix \mathbf{R} with random small entries. The shared instance is $(\mathbf{A}, \mathbf{Y} = \mathbf{AR} \bmod q)$, and the prover’s goal is to convince the verifier it knows a short \mathbf{R} satisfying $\mathbf{AR} = \mathbf{Y} \bmod q$.

The interactive protocol Π then executes as follows:

- The prover samples a short vector \mathbf{t} and sends $\alpha := \mathbf{A}\mathbf{t} \pmod q$.
- The verifier responds by sending a random vector \mathbf{c} with small entries.
- The prover responds with $\mathbf{z} := \mathbf{t} + \mathbf{R}\mathbf{c}$.
- The verifier accepts if $\mathbf{A} \cdot \mathbf{z} = \alpha + \mathbf{Y} \cdot \mathbf{c} \pmod q$ and \mathbf{z} is short.

As in [40], this interactive protocol is average-case sound under the SIS assumption. We now analyze the non-interactive protocol $\Pi_{\text{FS},\mathbf{h}}$ for a (vector-valued) Fiat-Shamir hash function \mathbf{h} . A malicious prover attacking the average-case soundness of $\Pi_{\text{FS},\mathbf{h}}$ must solve the following problem.

- **Input:** Random matrices (\mathbf{A}, \mathbf{Y}) and the description of a (vector-valued) hash function \mathbf{h} .⁴
- **Output:** Vectors α, \mathbf{z} such that $\mathbf{A} \cdot \mathbf{z} = \alpha + \mathbf{Y} \cdot \mathbf{h}(\alpha) \pmod q$ and \mathbf{z} is short.

Our main insight is that this problem is provably hard for a fixed Fiat-Shamir hash function \mathbf{h} if simple information-theoretic conditions are satisfied.

Theorem 7. *Suppose \mathbf{h} satisfies the following properties:*

1. \mathbf{h} produces “short” output, i.e., the entries are small relative to the modulus
2. α is a linear function of $\mathbf{h}(\alpha)$, i.e. there exists a matrix \mathbf{G} such that for all α , $\mathbf{G} \cdot \mathbf{h}(\alpha) = \alpha \pmod q$.

Then, $\Pi_{\text{FS},\mathbf{h}}$ is one-time (average-case) sound.

Theorem 7 can be proved as follows. If the condition in Theorem 7 are satisfied, then the relation $\mathbf{A} \cdot \mathbf{z} - \alpha - \mathbf{Y} \cdot \mathbf{h}(\alpha) = \mathbf{0} \pmod q$ checked by the verifier can be rewritten as

$$[\mathbf{A} \parallel \mathbf{Y} + \mathbf{G}] \cdot \begin{bmatrix} \mathbf{z} \\ -\mathbf{h}(\alpha) \end{bmatrix} = \mathbf{0} \pmod q. \tag{1}$$

Since \mathbf{A}, \mathbf{Y} are (statistically) uniformly random and $\mathbf{z}, \mathbf{h}(\alpha)$ are short, a malicious prover outputting α, \mathbf{z} is solving SIS for the random matrix $[\mathbf{A} \parallel \mathbf{Y} + \mathbf{G}]$.

A simple concrete instantiation of \mathbf{h} is the bit-decomposition function that maps (vectors of) \mathbb{Z}_q elements to (the concatenation of) their bit decomposition in $\{0, 1\}^{\lceil \log q \rceil}$ (also called $\mathbf{G}^{-1}(\cdot)$ in the lattice literature). The corresponding \mathbf{G} is the “powers-of-two” gadget matrix of Micciancio-Peikert [44].

Connections to Lattice Signatures from Lattice Trapdoors. Interestingly, it turns out the honest prover algorithm of the rejection sampling-based protocol *exactly* matches the trapdoor preimage sampling algorithm of Lyubashevsky-Wichs [41] using a Micciancio-Peikert trapdoor [44]. This can be seen by considering Eq. (1), which implies that the transcript of the protocol gives a short preimage of $\mathbf{0}$ of a matrix with a Micciancio-Peikert trapdoor (here \mathbf{R}). Average-case soundness

⁴ \mathbf{Y} is technically sampled as $\mathbf{A} \cdot \mathbf{R}$ for some a “short” matrix \mathbf{R} , but parameters are set so that \mathbf{Y} is statistically close to uniform.

implies that this should be hard to do without knowledge of \mathbf{R} (further using that $[\mathbf{A} \parallel \mathbf{A}\mathbf{R} + \mathbf{G}]$ looks uniformly random over the randomness of \mathbf{R}), and witness-indistinguishability implies that the preimage sampling algorithm reveals no more information about the trapdoor \mathbf{R} .

In fact, our protocol shows the connection between seemingly orthogonal paths to obtain signatures from lattice-based assumptions: one relying on lattice trapdoors and trapdoor preimage sampling [30, 41, 44] and another through Fiat-Shamir [38–40]. The lattice signature schemes constructed from lattice trapdoors [30, 41, 44] can actually be *derived* by applying the Fiat-Shamir heuristic (with aborts) using the bit-decomposition function (namely $\mathbf{G}^{-1}(\cdot)$) as the hash function to Lyubashevsky’s three-message identification scheme [40]. Let us start by describing the signature scheme for signing a short random message $\mathbf{v} \in \mathbb{Z}_q^n$. The Fiat-Shamir hash function takes as input the first message α from the protocol, and the message \mathbf{v} , and outputs

$$h(\alpha, \mathbf{v}) = \mathbf{G}^{-1}(\alpha - \mathbf{v}).$$

The signature consists of the challenge $\mathbf{c} = \mathbf{G}^{-1}(\alpha - \mathbf{v})$ and \mathbf{z} from the third message of the protocol. The verifier of the signature takes \mathbf{v} and its signature, and accepts if $\mathbf{A} \cdot \mathbf{z} = \alpha + \mathbf{Y} \cdot \mathbf{c} \pmod q$ and \mathbf{z} is short, that is:

$$[\mathbf{A} \parallel \mathbf{G} + \mathbf{Y}] \begin{bmatrix} \mathbf{z} \\ -\mathbf{c} \end{bmatrix} = \mathbf{v} \pmod q. \quad (2)$$

We now argue that this gives a signature scheme for random (short) messages, where the adversary can receive signature of random messages, and seeks to forge a signature for a random message given by the challenger. To handle signing queries, one can sample (\mathbf{z}, \mathbf{c}) , and set the message as $\mathbf{v} = [\mathbf{A} \parallel \mathbf{G} + \mathbf{Y}] \begin{bmatrix} \mathbf{z} \\ -\mathbf{c} \end{bmatrix}$.

Then, the hardness of signing a random message \mathbf{v} is then equivalent to breaking the SIS problem for a random target \mathbf{v} . To sign an arbitrary long message μ , we replace \mathbf{v} in the previous protocol by $H(\mu)$ where H is a random oracle. This exactly recovers the trapdoor-based lattice signatures [30, 41, 44] in the random oracle model. We stress that here, the only purpose of the random oracle is to compress the message (in a hash-and-sign manner), as opposed to collapse an interactive protocol. In particular the Fiat-Shamir hash function is still the non-cryptographic \mathbf{G}^{-1} function.

2.2 Fiat-Shamir for Schnorr in the Generic Group Model

The following section on the generic group model (GGM) contains a number of technical arguments, designed to motivate and provide intuition for our group-based results. We provide a roadmap for the discussion:

1. First we explain why Fiat-Shamir for Schnorr is secure in the (plain) GGM, even for simple, information-theoretic hash functions. We start with the case of “no-message” signatures (non-interactive identification) and then extend

our reasoning to handle messages and signing queries.

We remark that our security claims for Schnorr in the *plain* GGM could have been proven using prior analysis of [46]. However, we have two reasons for “re-doing” the analysis here: (1) our goal is to provide clear intuition tailored to *information-theoretic* Fiat-Shamir hash functions, and (2) our analysis will readily extend to the auxiliary-input setting, which we motivate next.

2. We will demonstrate that for Schnorr signatures, a (plain) GGM security proof does not capture a class of non-uniform attacks that work on *any concrete group*. In fact, we show that for common groups such as \mathbb{Z}_p^* , these attacks do not even require non-uniform advice.
3. We address these issues by extending our analysis to hold in the *auxiliary-input* GGM, albeit for a slightly more restricted class of Fiat-Shamir hash functions. We show this class still contains simple, information-theoretic hash functions, and we discuss potential implications of these results.

Non-Interactive Identification in the Generic Group Model. We begin by considering the classic Schnorr protocol for proving knowledge of a discrete logarithm. Recall that the protocol relies on a cryptographic group $G = \langle g \rangle$ of prime order p . The prover and verifier share an instance g^u for a random u known to the honest prover, and engage in the following interaction:

- The prover samples a random $r \leftarrow \mathbb{Z}_p$ and sends g^r .
- The verifier replies with a random $c \leftarrow \mathbb{Z}_p$.
- The prover sends $z = r + cu$.
- The verifier accepts if $g^z = (g^r)(g^u)^c$.

To build intuition, we will try to construct a (one-time secure) non-interactive identification scheme using a simple Fiat-Shamir hash function. In a moment, we will extend this (to handle messages and signing queries) to build full-fledged digital signatures.

For a Fiat-Shamir hash function h , a malicious prover for the non-interactive Schnorr protocol must solve the following problem.

- **Input:** A group description $G = (g, p)$, a hash function $h : G \rightarrow \mathbb{Z}_p$, and a random group element g^u .
- **Output:** g^r, z satisfying $g^z = (g^r)(g^u)^{h(g^r)}$.

We want to identify simple choices of h that make this problem hard in the GGM. However, it will be illuminating to instead identify which choices of h will make this problem *easy*.

This problem is clearly easy if h is a constant function, i.e. $h(g^x) = c$ for all g^x ; the malicious prover could always win by outputting $z = 0$ and $g^r = ((g^u)^c)^{-1} = g^{-uc}$. Taking this a step further, we can argue that for any constant $c \in \mathbb{Z}_p$, the hash function h should not output c on a $1/\text{poly}(\lambda)$ fraction of its inputs. Otherwise, a malicious prover can pick a random z and set $g^r = g^{-uc+z}$. Since g^r is distributed randomly, $h(g^r) = c$ holds with $1/\text{poly}(\lambda)$ probability, in which case z, g^{-uc+z} is a solution.

Put another way, as long as the min-entropy of h on a random input is $O(\log(\lambda))$, the above is a completely generic method (i.e. one that works on any cyclic group) for breaking the resulting non-interactive protocol.

It turns out that this simple class of h —those functions which, on random inputs, produce a low min-entropy output—are the *only* hash functions for which generic group algorithms (in the sense of Shoup [53]) exist to solve the above problem. That is, all hash functions h with super-logarithmic min-entropy can be proven to soundly compile non-interactive Schnorr in the GGM:

Theorem 8. *In the generic group model (GGM), the non-interactive Schnorr protocol is one-time secure provided $h(\cdot)$ on a random input has entropy $\omega(\log \lambda)$.*

Recall that in the generic group model, group elements g^x are replaced by labels $\sigma(x)$ where σ is a random injection from \mathbb{Z}_p to an exponentially-larger label space $[L]$ (say of size $\Omega(p^3)$, where p itself is a λ -bit prime). The attacker interacts with an oracle (who knows the truth table of σ) to perform honest group operations such as raising a group element to a known exponent, performing the group operation on any two group elements, and taking the inverse.

In this model, the only way an attacker can output a valid group label $\sigma(r)$ is to obtain this label from oracle queries (with overwhelming probability, any other label it might choose to output will not have a preimage). Furthermore, if the attacker is initialized with $\sigma(1), \sigma(u)$ for random $u \leftarrow \mathbb{Z}_p$, then any label it obtains from the oracle is of the form $\sigma(\alpha \cdot u + \beta)$, where α, β can be determined from prior oracle queries. In other words, the attacker must “know” α and β .

The attacker is trying to find z along with $\sigma(r)$ such that $z = r + u \cdot h(\sigma(r))$. But the attacker knows α and β such that $r = \alpha \cdot u + \beta$, so this equation can be written as $z = \alpha \cdot u + \beta + u \cdot h(\sigma(\alpha \cdot u + \beta))$. If $\alpha + h(\sigma(\alpha \cdot u + \beta)) \neq 0$, then the attacker can solve for u . However, this means the attacker has found a discrete log, which it can only do with negligible probability [53].

Therefore, it must be the case that $\alpha + h(\sigma(\alpha \cdot u + \beta)) = 0$. However, the poly-query attacker only learns $\sigma(\alpha \cdot u + \beta)$ for poly-many choices of (α, β) , and for each distinct choice of (α, β) , the resulting label $\sigma(\alpha \cdot u + \beta)$ is random. h evaluated on a random input has min-entropy $\omega(\log(\lambda))$, so the probability $\alpha + h(\sigma(\alpha \cdot u + \beta)) = 0$ holds is negligible; a union bound over the polynomially-many (α, β) oracle queries completes the argument.

Schnorr Signatures in the Generic Group Model. We now consider a slightly more difficult task: compiling Schnorr’s identification protocol into a digital signature scheme with existential unforgeability against chosen-message attacks (EUF-CMA security).

Note that the semantics of the hash function itself are now different: the standard Fiat-Shamir compiler for signatures takes as input a message $m \in \mathcal{M}$ to be signed (in addition to the first message of the interactive protocol), i.e. $h : G \times \mathcal{M} \rightarrow \mathbb{Z}_p$. For the purposes of this technical overview, we will restrict

to the case where \mathcal{M} is a $\text{poly}(\lambda)$ -size set.⁵ We stress that a restriction to only signing “short” messages will be crucial to the following discussion.

Furthermore, the EUF-CMA security experiment requires security in the presence of an unbounded number of signing queries. So the EUF-CMA attacker must solve following task:

- **Input:** A group description $G = (g, p)$, a hash function $h : G \times \mathcal{M} \rightarrow \mathbb{Z}_p$, and a random group element g^u .
- **Oracle Queries:** The attacker is free to make an unbounded number of queries to a signing oracle who knows u . It submits any $m \in \mathcal{M}$, the signing oracle samples a random $r \leftarrow \mathbb{Z}_p$, computes $z = r + h(g^r, m) \cdot u$, and returns the signature (g^r, z) .
- **Output:** Any $(m^*, (g^{r^*}, z^*))$ where $m^* \in \mathcal{M}$ satisfying $g^{z^*} = (g^{r^*})^{h(g^{r^*}, m^*) \cdot u}$ that was not the result of a signing query.

We would like to identify a class of hash functions h for which this problem is hard, and as in the previous section, we will start by identifying choices of h that make this problem *easy*.

Suppose that h has the following *undesirable* property: for some choice of $m \in \mathcal{M}$, the random variable obtained by sampling random $g^r \leftarrow G$ and outputting $h(g^r, m)$ has min-entropy $O(\log \lambda)$. In this case, breaking EUF-CMA security can be done efficiently without any signing queries. Let $c \in \mathbb{Z}_p$ be such that $h(g^r, m) = c$ holds with noticeable probability (guaranteed to exist by the low min-entropy property). The attack is to a uniformly random value $z \leftarrow \mathbb{Z}_p$, and then compute $g^r = g^{-uc+z}$. Since g^r is randomly distributed, then $h(g^r, m) = c$ with noticeable probability, and the resulting (g^r, z) constitutes a valid signature on m . To prevent this attack, we must require that for all $m \in \mathcal{M}$, the random variable $h(g^r, m)_{g^r \leftarrow G}$ has min-entropy $\omega(\log \lambda)$.

Another *undesirable* property of h is the following: suppose for some choice of distinct $m, m' \in \mathcal{M}$, the random variable $(\chi_{h(g^r, m)=h(g^r, m')})_{g^r \leftarrow G}$ (where $\chi_{x=y}$ is the indicator function that equals 1 if $x = y$ and 0 otherwise) has noticeable expected value, i.e. $h(g^r, m) = h(g^r, m')$ occurs with noticeable probability. If h satisfies this property, there is a straightforward attack using one signing query: the attacker queries on m , learns a random valid signature (g^r, z) , and then submits $(m', (g^r, z))$ as its forgery. Since the signing oracle provides a randomly generated valid signature (i.e. g^r is random in G), the Fiat-Shamir challenge for the m and m' executions will be identical with noticeable probability, meaning the signature (g^r, z) for m is a valid signature for m' with noticeable probability. To prevent this attack, we must require that for all distinct $m, m' \in \mathcal{M}$, the random variable $(\chi_{h(g^r, m)=h(g^r, m')})_{g^r \leftarrow G}$ has negligible expectation.

⁵ This restriction can in fact be relaxed somewhat, but our positive statements for information-theoretic Fiat-Shamir hash functions in the generic group model will crucially rely on $|\mathcal{M}|/p$ being negligible in λ .

To recap, we have the following *minimum* requirements on h :⁶

1. For all $m \in \mathcal{M}$, we the min entropy of $h(g^r, m)_{g^r \leftarrow G}$ is $\omega(\log \lambda)$.
2. For all distinct $m, m' \in \mathcal{M}$, we have $E_{g^r \leftarrow G}[(\chi_{h(g^r, m)=h(g^r, m')})] \leq \text{negl}(\lambda)$.

It turns out that these minimum requirements on h are sufficient to guarantee EUF-CMA security of Schnorr in the GGM:

Theorem 9. *Suppose $\mathcal{M} \subset \mathbb{Z}_p$ and $|\mathcal{M}| = \text{poly}(\lambda)$. Let $h : G \times \mathcal{M} \rightarrow \mathbb{Z}_p$ be any function satisfying conditions (1) and (2) above. Then the resulting Schnorr signature scheme is EUF-CMA secure in the generic group model.*

We first note that our proof of Theorem 8 implies that an attacker cannot generate a valid forgery before it has received any signing queries. That is, given $\sigma(u)$, the attacker cannot output $(m^*, (\sigma(r^*), z^*))$ where $m^* \in \mathcal{M}$ and $z^* = r^* + h(\sigma(r^*), m^*) \cdot u$. To see this, note that for any fixed m , the hash function $h(\cdot, m)$ satisfies the same min-entropy property required for non-interactive identification (by condition (1) on h). A union bound over \mathcal{M} implies the attacker cannot provide a forgery for any m .

Given this analysis, we prove Theorem 9 in two steps.

- **Step 1: Generate signing queries without knowledge of u .** In this step, we write down a hybrid experiment in which the adversary’s view has *no explicit dependence* on the discrete logarithm u . We accomplish this by instead *programming* the group oracle.

In more detail, when signing queries are answered honestly, the adversary receives $(\sigma(r), r + u \cdot h(\sigma(r), m))$. However, these signing queries can be *simulated* in the following way:

- Sample a random label $\ell \leftarrow [L]$
- Sample a random exponent $z \leftarrow \mathbb{Z}_p$.
- Program the value $\sigma(z_i - x \cdot h(\ell, m)) = \ell$. If the oracle σ was already programmed at ℓ , abort.
- Output the signature (ℓ, z, m)

Moreover, this gives us an *implicit representation* of the group element corresponding to label ℓ as a *publicly known* linear combination of g^u and g , namely, $(g^z \cdot (g^u)^{-h(\ell, m)})$. These group elements will all be distinct with high probability over the choice of u .

⁶ This is the characterization for the case $|\mathcal{M}| = \text{poly}(\lambda)$. For larger message spaces (that still satisfy $|\mathcal{M}|/p \leq \text{negl}(\lambda)$), the requirements are mildly strengthened: we require that (1) for all targets $c \in \mathbb{Z}_p$, the probability over a random choice of r that $h(g^r, m) = c$ for any m is negligible, and that for any $m \in \mathcal{M}$, the probability over a random choice of r that $h(g^r, m') = h(g^r, m)$ for any m' is negligible (i.e., we reversed an order of quantifiers in each requirement). These are exactly information-theoretic analogues of the RPP and RPSP properties defined in [46].

Essentially, this simulated experiment is indistinguishable from the real security game as long as the programmed values $\sigma(z_i - u \cdot h(\ell, m))$ do not contradict any of the adversary's previous queries to the group oracle. One can show that the probability of this is negligible because of the randomness of u according to the adversary's view. This is effectively an invocation of the generic group hardness of computing discrete logs.

– **Step 2: Invoke the statistical properties of h .** Now that we have simulated all of the signature queries, we consider a potential forgery $(\sigma(r^*), z^* = r^* + u \cdot h(\sigma(r^*), m^*), m^*)$ and break into two cases.

- **Case 1: $\ell^* := \sigma(r^*)$ matches one of the signing queries.** In this case, we claim that a forgery allows us to compute the discrete logarithm u . Indeed, this is because we have a signing query equation of the form

$$z = r^* + h(\ell^*, m)$$

and a forgery equation of the form

$$z^* = r^* + h(\ell^*, m^*)u.$$

Moreover, the two hash values $(h(\ell^*, m), h(\ell^*, m^*))$ must be distinct because (1) the marginal distribution on ℓ^* is random, and (2) we assumed that for a random ℓ^* , there will not exist an h -collision with prefix ℓ^* .

- **Case 2: ℓ^* does not match any signing query.** In this case, we also claim that a forgery allows us to compute the discrete logarithm u . Indeed, the forgery equation

$$z^* = r^* + h(\ell^*, m^*)u$$

along with the adversary's implicit representation of the exponent

$$r^* = \alpha + \beta u$$

(which follows from the fact that the adversary's view can be computed generically given only g^u) implies that

$$z^* = \alpha + (\beta + h(\ell^*, m^*))u.$$

Then, either $\beta + h(\ell^*, m^*) \neq 0$, in which case the adversary can indeed compute u , or $\beta + h(\ell^*, m^*) = 0$. We claim that the high min-entropy of $h(\ell, m)$ for random ℓ implies that this event is unlikely. Indeed, ℓ^* must have been obtained by *some* group oracle query, so this follows by a union bound over all group oracle queries made by the adversary.

This completes our proof sketch of Theorem 9.

Preprocessing Attacks. We next show how the [46] characterization of Schnorr signature security in the GGM fails to capture security in concrete groups. Since the attacks that we discover fall into the framework of the auxiliary-input GGM [18, 55], we then analyze Schnorr signatures in this stronger adversary model.

We first describe an attack in the case of Schnorr signatures for short messages, using the hash function $h(g^r, m) = g^r + m \pmod{p}$ over the group⁷ $G = \mathbb{Z}_p^\times$. We showed above that this signature scheme is secure in the generic group model, but we will nonetheless give an attack over \mathbb{Z}_p^\times .

In order to have a well-specified protocol, we need to fix a mapping $\text{Int} : G \rightarrow \mathbb{Z}$ from group elements to integers. For simplicity, we choose our mapping so that $R \in \mathbb{Z}_p^\times$ maps to the unique integer $a \in [-\frac{p-1}{2}, \frac{p-1}{2}]$ such that $R \equiv a \pmod{p}$.

The attack proceeds as follows: we are given a random group element g^u and want to output m, g^r, z satisfying $g^z = (g^r)(g^u)^{\text{Int}(g^r)+m}$. We do this by picking r, m such that $\text{Int}(g^r) + m = 0 \pmod{p-1}$ and then setting $r = z$. So, for example, if the message space \mathcal{M} contains $m = p-2$, then we can pick $r = 0$, so that $g^r \equiv 1 \pmod{p}$ and $1 + p - 2 \equiv 0 \pmod{p-1}$. This choice is by no means special; if $1 \in \mathcal{M}$, then we can pick $r = \frac{p-1}{2}$ and obtain another forgery.

This strategy readily generalizes to groups beyond \mathbb{Z}_p^\times : for a cyclic group G of order p , all that is required to produce a forgery is knowledge of an exponent $r \in \mathbb{Z}_p$ and a message $\mu \in \mathcal{M} \subset \mathbb{Z}_p$ such that $\text{Int}(g^r) = -\mu \pmod{p}$. It also generalizes to the case of full Schnorr signatures over G , using hash functions of the form $h(g^r, m) = \text{Int}(g^r) + H(m)$ for a collision-resistant hash function H . One can check that the hash function (family) h satisfies the hypotheses of [46], so Schnorr signatures using h are secure in the GGM. However, if G has a known equation of the form

$$\text{Int}(g^r) = -\mu,$$

and H additionally satisfies $H(0) = \mu$ (which can be arranged without sacrificing collision resistance by hard-coding this value into a hash function H whose range excludes μ), then again (r, r) is a valid signature. Thus, we see that for every group G with some hard-coded equation $\text{Int}(g^r) = -\mu$, there exists a hash family h satisfying the [46] hypotheses which leads to an *insecure* instantiation of Schnorr signatures.

We now observe that one can view this attack as an attack in the *auxiliary-input generic group model*. The Aux-Input GGM is the following adversary model for some problem \mathcal{P} over a group G .

- The adversary is given the description of a group G as a random injection from $G \rightarrow [L]$ (i.e., the adversary is given the full truth tables of the group operation).
- The adversary then stores S bits of information about this group G (and forgets everything else).
- The adversary then receives an instance of \mathcal{P} (as characterized by a security game with a challenger). As in the GGM, the adversary can also query the group oracle.

In other words, an aux-input GGM adversary is a GGM adversary that is augmented with some S bits of non-uniform advice about the group.

⁷ This group does not have prime order, but this detail is not relevant to our analysis.

Given this definition, it is easy to see that the attacks described above fall into the aux-input GGM. Indeed, as long as the adversary “remembers” one equation of the form $\text{Int}(g^r) = -\mu$ (of which many are guaranteed to exist), it will be able to execute an attack. Thus, one can view the attacks on \mathbb{Z}_p^\times and other groups as the result of the following three-step process:

- There exist attacks on the schemes above in the auxiliary-input GGM. This means that for every concrete group G , there exists a *non-uniform* attack on the scheme.
- In the case of specific groups such as \mathbb{Z}_p^\times , the non-uniform advice necessary to carry out the attack can be computed efficiently given the group description.

Security in the Aux-Input GGM. Given the existence of preprocessing attacks as above, in order to have confidence in the *concrete* security of a Schnorr signature scheme using hash family h , it is necessary to prove security in the auxiliary-input GGM.

Just as in the case of our GGM lower bounds, we give a characterization of hash functions (and hash function families) h that lead to secure Schnorr signatures in the auxiliary-input GGM. We state a special case of our theorem for the purposes of this overview; we refer to the full version for a more general statement.

Theorem 10. *Let $\mathcal{M} \subset \mathbb{Z}_p$ and $|\mathcal{M}|/\mathbb{Z}_p \leq \text{negl}(\lambda)$. Suppose the (keyed) Fiat-Shamir hash function $H_k : [L] \times \mathcal{M} \rightarrow \mathbb{Z}_p$ satisfies the following properties:*

- For any $m \in \mathcal{M}$, $h(g^u, m)$ has min-entropy $\log(|\mathcal{M}|) \cdot \log \lambda$ on a random $g^u \leftarrow G$.
- **Zero-avoidance:** For any (stateful, potentially unbounded) adversary \mathcal{A} :

$$\Pr [H_k(\ell, m) = 0 \mid \ell \leftarrow \mathcal{A}(1^\lambda), k \leftarrow \mathcal{K}, m \leftarrow \mathcal{A}(k)] \leq \text{negl}(\lambda);$$

Then Schnorr signatures with Fiat-Shamir hash function H_k are EUF-CMA secure in the AI-GGM against adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ with advice of size $S = \text{poly}(\lambda)$, $T = \text{poly}(\lambda)$ oracle queries, $Q = \text{poly}(\lambda)$ signing queries.

The first of the two hypotheses is the same as in Theorem 9; the second rules out the preprocessing attacks described above. Similarly to before, Theorem 10 says that once these attacks are avoided, no further attacks in the Aux-Input GGM exist.

We prove Theorem 10 using the framework of [18], who show a rough equivalence between the auxiliary-input GGM and an a priori weaker adversary model called the *bit-fixing GGM (BF-GGM)*. Informally, in the BF-GGM, instead of learning an arbitrary S bits of information about a random group G , the adversary can only remember the *labels* of P group elements (and their corresponding exponents with respect to the canonical generator). In [18], it is shown that for any (efficient and generic) challenger-adversary game, security in the AI-GGM follows from security in the (ostensibly weaker) BF-GGM with a slight loss in

parameters. We can apply this result directly to the soundness of Schnorr signatures, reducing our problem to proving a lower bound in the BF-GGM.

Now, we can conveniently extend all of our GGM analysis (i.e., the proof of Theorem 9 to apply in the BF-GGM (and therefore to the AI-GGM via [18])). The BF-GGM lower bound will look very similar to before:

- **Step 1: Generate signing queries without knowledge of u .** We simulate signing queries in exactly the same way as before. Some care is required to argue that indistinguishability still holds, because the adversary additionally has access to a short list of hard-coded group labels.
- **Step 2: Invoke the statistical properties of h .** We again consider a potential forgery $(\sigma(r^*), z^* = r^* + h(\sigma(r^*), m^*)u, m^*)$. This time, we break into *three* cases:
 - **Case 0: ℓ^* appears in the adversary’s auxiliary information.** This case is unique to the BF-GGM setting; however, the forgery equation

$$z^* = r^* + h(\ell^*, m)u$$

allows us to solve for u unless $h(\ell^*, m) = 0$, which cannot happen (except with negligible probability) because we assumed that h was 0-avoiding.

- **Case 1: $\ell^* := \sigma(r^*)$ matches one of the signing queries.** This case matches our GGM analysis above.
- **Case 2: ℓ^* does not match any signing query.** This case also matches our GGM analysis above.

This completes our proof sketch of Theorem 10.

Application: (Candidate) Simple Schnorr Signatures. One takeaway of our analysis is that it *might* be possible that simple compilations of Schnorr signatures (for small message space) are secure. The appeal of such a signature scheme is that all of the operations are extremely simple, and can be implemented with random sampling and modular arithmetic. We stress that the only evidence we have for security is that this scheme resists *generic preprocessing attacks*, and that so far, we have been unable to leverage non-generic properties of \mathbb{Z}_p^\times to break this scheme. Further analysis of this simple scheme is beyond the immediate scope of this work, and we *strongly recommend* against considering this scheme “secure” unless it withstands significant cryptanalytic effort.

Construction 11. Consider the Schnorr signature scheme for group \mathbb{Z}_p^\times , where the Fiat-Shamir hash function has random $k \leftarrow \mathbb{Z}_q$, and outputs $g^r + m + k \pmod q$ on input (g^r, m) :

- Group: \mathbb{Z}_p^\times with a generator g of a cyclic subgroup of order q , where $p = 2q + 1$.
- Message space: Any subset $M \subset \mathbb{Z}_q$ of $\text{poly}(\lambda)$ size.
- Signing key: $sk \leftarrow \mathbb{Z}_q$.
- Verification key: (k, g^{sk}) where $k \leftarrow \mathbb{Z}_q$.
- Sign(sk, m): Sample $r \leftarrow \mathbb{Z}_q$. Let $z = r + (g^r + m + k) \cdot sk \pmod q$. Output (g^r, z) .
- Ver($vk, m, (g^r, z)$): Accept if $g^z = g^r \cdot (g^{sk})^{g^r + m + k} \pmod p$.

Extensions to Chaum-Pedersen and NIZKs for NP. Our analysis for Schnorr signatures in the AI-GGM easily extends to prove *semi-adaptive* soundness of the Chaum-Pedersen protocol for proving validity of a Diffie-Hellman tuple. As the security analysis is extremely similar to our analysis for Schnorr, we defer this result (and its implications for NIZKs for NP) to the full version.

2.3 Negative Results

In this section, we give a simple example of a negative result that we can prove using our methods. In particular, we consider an idealized variant of Blum’s Hamiltonicity protocol [8] in which the commitment scheme is instantiated with a random oracle.

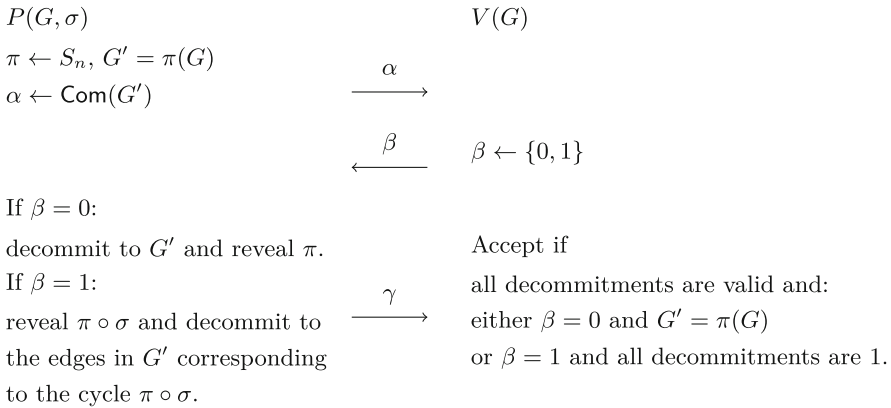


Fig. 1. The Zero Knowledge Proof System Π^{Blum} for Graph Hamiltonicity.

The Blum protocol $\Pi = \Pi^{\text{Blum}}$ is described in Fig. 1. For this example, we instantiate $\text{Com}(b; r) = \mathcal{O}(b, r)$ as an idealized bitwise commitment scheme in the random oracle model. Π then is repeated t times in parallel to obtain soundness error 2^{-t} .

At first glance, especially given our positive results for Schnorr and Chaum-Pedersen, one might hypothesize that since we have made the commitment scheme “super-secure”, Fiat-Shamir for Π^t might be instantiable with a simple hash function h . In fact, we show that even for this idealized variant of the Blum protocol, a (successful) Fiat-Shamir hash function h for this protocol necessarily satisfies a cryptographic security property.

As discussed earlier, there are two variants of this result. First, we give a polynomial-query attack on $\Pi_{\text{FS},h}^t$ for any hash function h that does not invoke the random oracle \mathcal{O} . Then, we extend this polynomial-query attack to a polynomial-time attack assuming the *easiness* of some computational problem depending on h .

To understand our attack, we first consider an “obviously broken” choice of hash function h : define $h(\alpha_1, \dots, \alpha_t) = (f(\alpha_1), \dots, f(\alpha_t))$ to be a fixed function

applied to each commitment separately. This corresponds to a parallel repetition of $\Pi_{\text{FS},f}$, which is the application of Fiat-Shamir to a protocol with constant soundness error. We know that such a non-interactive protocol is unsound via a *reset attack*: given an instance G , it is possible to prepare a commitment α_1 that can successfully answer either a “0” challenge or a “1” challenge. Therefore, if α_1 is prepared to answer the challenge b (for a uniformly random bit b), we have that $f(\alpha_1) = b$ with probability $1/2$ (since α_1 hides b) and so after an expected constant number of string commitment queries, we obtain an accepting transcript $(\alpha_1, b_1, \gamma_1)$ for the first repetition. This can be done for each “slot”, giving a polynomial-query break of soundness for the overall protocol.

To rephrase the attack, for our example choice of h , if one prepares enough “fake commitments” $\{\alpha_1^{(i)}\}, \{\alpha_2^{(i)}\}, \dots, \{\alpha_t^{(i)}\}$ for each of the t repetitions, then with high probability, there exists a *combination* of the individual commitments that hashes to the “bad challenge” whose answer was generated along with the commitments. We show that the above argument generalizes to *all* hash functions h . The poly-query attack is as follows.

1. For $1 \leq i \leq t, 1 \leq \ell \leq q$, sample a random bit $y_\ell^{(i)} \leftarrow \{0, 1\}$ and sample message $\alpha_\ell^{(i)}$: if $y_\ell^{(i)} = 0$, sample $\alpha_\ell^{(i)}$ as in the honest protocol, while if $y_\ell^{(i)} = 1$, and sample $\alpha_i^{(\ell)}$ as a commitment to a cycle graph.
2. Find $v \in [q]^t$ such that $h(\alpha[v]) = y[v]$. Abort if no such v exists.
3. Output $\alpha[v]$ as well as the necessary decommitments to $\alpha[v]$ (either the entire graph or just the edges in the cycle).

This constitutes a poly-query attack on the protocol $\Pi_{\text{FS},H}^t$ in the random oracle model as long as Step (2) has a solution with high probability over (α, y) . In the case $h = (f, \dots, f)$ as above, this condition follows immediately. We show in the full version that for *any* h , as long as $q = \omega(t)$, Step (2) has a solution with high probability over (α, y) .

To obtain a (conditional) polynomial-time attack on the protocol, we note that if the solution to the problem in Step (2) can be found *efficiently*, then the above attack can be implemented in polynomial time.

Crucially, the above analysis generalizes well because the computational problem in Step (2) does not depend on the protocol. We accomplish this by reducing breaking the soundness of $\Pi_{\text{FS},h}^t$ to solving a “mix-and-match” problem of the following form: given many strings $\{\alpha_\ell^{(i)}\}$ (q strings for each slot) which are each associated with a random bit $b_\ell^{(i)}$, find a concatenation $\alpha[v]$ of t different $\alpha_\ell^{(i)}$ (one for each slot) such that $h(\alpha[v]) = b[v]$ (the corresponding combination of bits). This motivates our definition of “mix-and-match resistance” (see the full version), a security property which captures the analogous problems for a wide class of protocols Π .

While the analysis above is tailored to (parallel repeated) Π^{Blum} , it turns out that the argument only relies on a couple of (basic) properties of the protocol, namely:

- Given a challenge β , it is possible to sample a (pseudorandom) first message α along with an accepting response γ for α , even when the statement x is false. This property is used to construct a mix-and-match problem in our attack, and essentially follows from an *honest-verifier zero knowledge* property of the protocol.
- The protocol is obtained by applying parallel repetition to a protocol with *polynomial-size* challenge space. This independence property is enough to guarantee that the “mix-and-match” problem information-theoretically has a solution.

We refer the reader to the full version for more details on the extent to which the result generalizes.

Acknowledgments. We thank Brynmor Chapman, Justin Holmgren, Akshayaram Srinivasan, and Daniel Wichs for many helpful discussions. Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing in Spring 2020.

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_21
2. Ben-Sasson, E., et al.: Computational integrity with a public random string from quasi-linear PCPs. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 551–579. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_19
3. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl, July 2018
4. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018). <https://eprint.iacr.org/2018/046>
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 701–732. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_23
6. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_4
7. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2
8. Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, vol. 1, p. 2. Citeseer (1986)

9. Brakerski, Z., Koppula, V., Mour, T.: NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 738–767. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_26
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS, pp. 97–106. IEEE Computer Society Press, October 2011
11. Canetti, R., et al.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC, pp. 1082–1090. ACM Press, June 2019
12. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 91–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_4
13. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC, pp. 209–218. ACM Press, May 1998
14. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
15. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_7
16. Chen, Y., Lombardi, A., Ma, F., Quach, W.: Does Fiat-Shamir require a cryptographic hash function? Cryptology ePrint Archive, Report 2020/915 (2020). <https://eprint.iacr.org/2020/915>
17. Ciampi, M., Parisella, R., Venturi, D.: On adaptive security of delayed-input sigma protocols and Fiat-Shamir NIZKs. In: Galdi, C., Kolesnikov, V. (eds.) SCN 2020. LNCS, vol. 12238, pp. 670–690. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57990-6_33
18. Coretti, S., Dodis, Y., Guo, S.: Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 693–721. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_23
19. Corrigan-Gibbs, H., Kogan, D.: The discrete-logarithm problem with preprocessing. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 415–447. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_14
20. Couteau, G., Hofheinz, D.: Designated-verifier pseudorandom generators, and their applications. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 562–592. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_20
21. Couteau, G., Katsumata, S., Ursu, B.: Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 442–471. Springer, Heidelberg (May 2020)
22. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055717>
23. Damgård, I.: On sigma-protocols. Lecture Notes, Faculty of Science, Department of Computer Science, Aarhus University (2010)

24. Dent, A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 100–109. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_6
25. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 3–32. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_1
26. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.* **29**(1), 1–28 (1999)
27. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
28. Fischlin, M.: A note on security proofs in the generic model. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 458–469. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_35
29. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 169–178. ACM Press, May/June 2009
30. 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
31. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS, pp. 464–479. IEEE Computer Society Press, October 1984
32. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: 14th ACM STOC, pp. 365–377. ACM Press, May 1982
33. Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: one-way product functions and their applications). In: Thorup, M. (ed.) 59th FOCS, pp. 850–858. IEEE Computer Society Press, October 2018
34. Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 224–251. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_8
35. Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 622–651. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_22
36. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC, pp. 723–732. ACM Press, May 1992
37. Lombardi, A., Vaikuntanathan, V.: Fiat-Shamir for repeated squaring with applications to PPAD-hardness and VDFs. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 632–651. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_22
38. Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 162–179. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_10
39. Lyubashevsky, V.: Fiat-Shamir with aborts: applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_35

40. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
41. Lyubashevsky, V., Wichs, D.: Simple lattice trapdoor sampling from a broad class of distributions. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 716–730. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_32
42. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
43. Micali, S.: Computationally sound proofs. *SIAM J. Comput.* **30**(4), 1253–1298 (2000)
44. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
45. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. *Math. Notes* **55**(2), 165–172 (1994). <https://doi.org/10.1007/BF02113297>
46. Neven, G., Smart, N.P., Warinschi, B.: Hash function requirements for Schnorr signatures. *J. Math. Cryptol.* **3**(1), 69–87 (2009)
47. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_3
48. Peikert, C.: A decade of lattice cryptography. *Found. Trends Theoret. Comput. Sci.* **10**(4), 283–424 (2016)
49. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 89–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_4
50. Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 252–265. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0034852>
51. Quach, W., Rothblum, R.D., Wichs, D.: Reusable designated-verifier NIZKs for all NP from CDH. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 593–621. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_21
52. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22
53. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
54. Stern, J., Pointcheval, D., Malone-Lee, J., Smart, N.P.: Flaws in applying proof methodologies to signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 93–110. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_7
55. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_12
56. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy, pp. 926–943. IEEE Computer Society Press, May 2018



Composition with Knowledge Assumptions

Thomas Kerber^(✉), Aggelos Kiayias, and Markulf Kohlweiss

The University of Edinburgh and IOHK, Edinburgh, Scotland
papers@tkerber.org, {akiayias,mkohlwei}@ed.ac.uk

Abstract. Zero-knowledge succinct non-interactive arguments (zk-SNARKs) rely on *knowledge assumptions* for their security. Meanwhile, as the complexity and scale of cryptographic systems continues to grow, the composition of secure protocols is of vital importance. The current gold standards of composable security, the Universal Composability and Constructive Cryptography frameworks cannot capture knowledge assumptions, as their core proofs of composition prohibit white-box extraction. In this paper, we present a formal model allowing the composition of knowledge assumptions. Despite showing impossibility for the general case, we demonstrate the model’s usefulness when limiting knowledge assumptions to few instances of protocols at a time. We finish by providing the first instance of a simultaneously succinct and composable zk-SNARK, by using existing results within our framework.

1 Introduction

Knowledge assumptions, a class of non-falsifiable assumptions, are often used in cases where both succinctness and extractability are required. Perhaps the most notable modern usage is in zk-SNARKs [12, 18, 20–22, 30, 33], which typically rely on either a knowledge-of-exponent assumption [13], the Algebraic Group Model (AGM) [17], or the even stronger Generic Group Model (GGM) [34].

The idea of utilising additional assumptions for extraction extends outside of what it traditionally considered a “knowledge assumption” to extractable functions, notably extractable one-way functions [9, 10], and extractable hash functions [4]. Arguably, one of the main benefits of the random oracle model, one of the most common “non-standard” assumptions, is to provide extractability.

The typical statement of these assumptions is that for every adversary there exists a corresponding extractor, such that when both are given the same inputs and randomness, the extractor can provide meaningful information about how the output of the adversary was created. In the Algebraic Group Model, for instance, the extractor will show how to represent the adversaries output as powers of input group elements, and in extractable hash functions it will provide a preimage to the output hash.

This is formalised as a security game, which is then assumed to hold axiomatically. The existence of the extractor may be used in a security proof to demonstrate the existence of a preimage. At the same time, a one-wayness property can

be asserted, with this differing subtly from extraction in that an adversary to one-wayness does not have access to the input and randomness to extract from. This methodology has seen success in proving the security of various interesting primitives, such as non-malleable codes [27], and SNARKs.

Proving these primitives' security under composition would typically involve using one of a number of off-the-shelf compositional frameworks, such as Universal Composability [8] or Constructive Cryptography [31], specifying an ideal behaviour for the primitive, and constructing a simulator which coerces the ideal behaviour to mimic that of the actual protocol. This simulator will naturally need to make use of the extraction properties, often to infer the exact ideal intent behind adversarial actions. It is in this that the conflict between extraction and compositional frameworks arises: As the extraction is white-box, the simulator requires the input of its counter-party – the environment, or distinguisher, of the simulation experiment. This cannot be allowed however, as it would give the simulator access to *all* information in the system¹, not just that of the adversary.

This conflict has been observed before in the literature, for instance in [28]. Often, the remedy is to extend the original protocol with additional components to enable the simulator to extract “black-box”, i.e. without the original inputs. For example, the Fischlin transform [15] uses multiple queries to a random oracle to bypass the inability to extract from the commitment phase of an underlying Sigma protocol, which would allow using the simpler Fiat-Shamir transform [14] instead. C0C0 [28] extends zk-SNARKs with an encryption of the witness, and a proof of correctness of this encryption to a public key the simulator can control.

A primary downside of these approaches is that (witness) succinctness is usually lost – size being limited by the information-theoretic reality of black-box witness extraction. Thus C0C0 proofs are longer than their witnesses, and UC-secure commitments [11] are longer than the message domain. A secondary one is the necessity of adding an encryption to the implementation of a primitive where no decryption would be needed; in theory this is harmless but in practice, it adds complexity without a functional purpose (beyond its usefulness in the security argument).

The above limitation can often be bypassed by using a local random oracle, as this *does* permit extraction. Restricting the model to allow the adversary to perform only specific computations on knowledge-implying objects, could be one way to generalise this approach. Just as a random oracle functionality would abstract over extractable hash functions, a generic group functionality would abstract over knowledge of exponent type assumptions. This would constitute a far stronger assumption however, running counter to recent developments to relax assumptions, such as the Algebraic Group Model [17], which aim for a more faithful representation of knowledge assumptions.

Our contributions. We present the first composability framework that can overcome the above limitations in terms of implementation complexity and

¹ Recall that the simulator *is* the ideal-world adversary, and should by definition not have access to secrets the distinguisher holds.

succinctness while being reasonably accommodating to more realistic models of computation compared to the random oracle model. In terms of applications, our approach suggests a viable direction for the composable design of SNARKs, a topic of current high interest due to their application in blockchain protocols (with prominent examples including privacy-preserving blockchains, such as [2, 26], blockchain interoperability [19] and scalability [16]), without sacrificing their succinctness, something infeasible with previous compositional approaches.

In more details, our work builds on the Algebraic Group Model approach and explores its consequences for composition. Our contributions are two-fold: We distill a notion of knowledge assumptions suitable for composable analyses, and present a framework allowing their usage in composable security proofs.

First, we define the concept of knowledge-respecting distinguishing environments, which we will call distinguishers, to be consistent with the terminology of Constructive Cryptography. We use the Constructive Cryptography framework [31] as an orientation point for this work, due to its relative simplicity compared to the many moving parts of UC [8], making it easier to re-establish composition after making sweeping changes to the model, as we do in this paper.

Similar to an algebraic algorithm, distinguishers in our model need to explain how they computed each knowledge-implying object they produce. We show how to extend a compositional framework by giving the simulator access to these explanations. For this purpose, we attach a type system to messages sent in the composition framework, which can mark which parts of messages imply further knowledge. The knowledge assumption is used to transform individual nodes in the network into corresponding nodes which also output this implied knowledge to a repository, which the simulator has oracle access to.

We re-establish well-known composition results with support for typed messages, and sets of valid distinguishers. The latter constrains the generality of the composition, but is what enables the usage of knowledge assumptions, as they require the distinguishers to be well-behaved.

Our second contribution investigates under which conditions it is reasonable to assume knowledge-respecting distinguishers. To this end, we define stronger versions of knowledge assumptions that depend on auxiliary and knowledge-implying inputs. These assumptions suffice to extend a distinguisher with an extractor providing said explanations.

Within this setting we are able to establish not only an impossibility result on full general composition, but more interestingly a positive result on the composition of systems relying on different knowledge assumptions. Intuitively: You can use a knowledge assumption only once, or you need to ensure the various uses do not interfere with each other (specifically, the simulators of both invocations cannot provide any advantage due to extraction, as shown in the example in Sect. 5). This result has the immediate effect of enabling the usage of primitives relying on knowledge assumptions in larger protocols – provided the underlying assumption is not used in multiple composing proofs.

We demonstrate the power of the framework by presenting a composable NIZK, which can be realised by any extractable zk-SNARK scheme with simulation extractability relying on the AGM. Notably, this is true of Groth's

zk-SNARK [1, 20]. To our knowledge, this is the first time a SNARK has been demonstrated to be composable secure without costly modifications to add black-box extraction. We additionally demonstrate that this may be combined with a protocol to securely instantiate an updateable reference string, when used with SNARKs supporting this, demonstrating that despite general-case impossibility, special composition cases that are highly relevant to current applications are provable within the framework.

2 Modelling Knowledge Assumptions

We formally define knowledge assumptions over a type of *knowledge-implying objects* X . When an object of the type X is produced, the assumption states that whoever produced it must know a corresponding witness of the type W . The *knowledge of exponent* assumption is an example of this, where X corresponds to pairs of group elements, and W is an exponent. A relation $\mathcal{R} \subseteq X \times W$ defines which witnesses are valid for which knowledge-implying objects.

In the case of the knowledge of exponent assumption, it roughly states that given a generator, and a random power s of the generator, the only way to produce a pair of group elements, where one is the s th power of the other, is to exponentiate the original pair, and in so doing implying knowledge of this exponent. There is one extra item needed: The initial exponent s needs to be sampled randomly. Indeed, this is true for *any* knowledge assumption: The all-quantification over potential distinguishers implies the existence of distinguishers which “know” objects in X without knowing their corresponding witness. To avoid this pre-knowledge, we assume X itself is randomly selected at the start of the protocol. For this purpose, we will assume a distribution init , which given a source of public randomness (such as a global common random string), produces *public parameters* pp , which parameterise the knowledge assumption. In the case of knowledge of exponent, this needs to sample an exponent s , and output the pair (g, g^s) . For this particular setup, public randomness is insufficient.

Beyond this, users do not operate in isolation: If Alice produces the pair (g^x, g^{xs}) , knowing x and transmits this to Bob, he can produce (g^{xy}, g^{xys}) *without* knowing xy . This does not mean that the knowledge assumption does not hold, however it is more complex than one might originally imagine: One party can use knowledge-implying objects from another user as (part of) their own witnesses. Crucially this needs to be limited to objects the user actually received: Bob *cannot* produce (g^{sy}, g^{s^2y}) for instance, as he never received (g^s, g^{s^2}) , and does not know s . This setting also lends itself more to some interpretations of knowledge assumptions than others. For instance, the classical knowledge-of-exponent assumption [13] does not allow linear combinations of inputs, while the t -knowledge-of-exponent assumption [23] does. When used composable, the latter is more “natural”, in much the same way that IND-CCA definitions of encryption fit better into compositional frameworks than IND-CPA ones, due to them already accounting for part of the composable interaction.

Definition 1 (Knowledge Assumption). A knowledge assumption \mathfrak{K} is defined by a tuple $(\text{init}, X, W, \mathcal{R})$ consisting of:

1. init , a private-coin distribution to sample public parameters pp from, which the others are parameterised by.
2. X_{pp} , the set of all objects which imply knowledge.
3. W_{pp} , the set of witnesses, where $\forall x \in X_{\text{pp}} : (\text{INPUT}, x) \in W_{\text{pp}}$.
4. $\mathcal{R}_{\text{pp}} : (I \subseteq X_{\text{pp}}) \rightarrow (Y \subseteq (X_{\text{pp}} \times W_{\text{pp}}))$, the relation new knowledge must satisfy, parameterised by input objects, where

$$\forall x, y \in X_{\text{pp}}, I \subseteq X_{\text{pp}} : (x, (\text{INPUT}, y)) \in \mathcal{R}_{\text{pp}}(I) \iff x = y \wedge x \in I.$$

\mathcal{R}_{pp} must be monotonically increasing: $\forall I \subseteq J \subseteq X_{\text{pp}} : \mathcal{R}_{\text{pp}}(I) \subseteq \mathcal{R}_{\text{pp}}(J)$.

The inclusion of (INPUT, x) in W_{pp} and \mathcal{R}_{pp} for all $x \in X_{\text{pp}}$ ensures that parties are permitted to know objects they have received as inputs, without needing to know corresponding witnesses. Importantly, this is possible *only* for inputs, and not for other objects. For each knowledge assumption \mathfrak{K} , the assumption it describes is in a setting of computational security, with a security parameter λ . Broadly, the assumption states that, for a restricted class of “ \mathfrak{K} -respecting” adversaries, it is possible to compute witnesses for each adversarial output, given the same inputs.

Assumption 1 (\mathfrak{K} -Knowledge). The assumption corresponding to the tuple $\mathfrak{K} = (\text{init}, X, W, \mathcal{R})$ is associated with a set of probabilistic polynomial time (PPT) algorithms, $\text{Resp}_{\mathfrak{K}}$. We will say an algorithm is \mathfrak{K} -respecting if it is in $\text{Resp}_{\mathfrak{K}}$. This set should contain all adversaries and protocols of interest. The \mathfrak{K} -knowledge assumption itself is then that, for all $\mathcal{A} \in \text{Resp}_{\mathfrak{K}}$, there exists a PPT extractor \mathcal{X} , such that:

$$\Pr \left[\begin{array}{l} \text{pp} \stackrel{r}{\leftarrow} \text{init}; \\ \exists I \subseteq X_{\text{pp}}, \text{aux} \in \{0, 1\}^* : \\ \text{Game } 1(\mathcal{A}_{r'}, \mathcal{X}_{r'}, \text{pp}, I, \text{aux}) \end{array} \right] \leq \text{negl}(\lambda),$$

where $\mathcal{A}_{r'}$ and $\mathcal{X}_{r'}$ are \mathcal{A} and \mathcal{X} supplied with the same random coins r' (as such, they behave deterministically within Game 1).

While it is trivial to construct adversaries which are not \mathfrak{K} -respecting by encoding knowledge-implying objects within the auxiliary input, these trivial cases are isomorphic to an adversary which *is* \mathfrak{K} -respecting, and which receives such encoded objects directly. We therefore limit ourselves to considering adversaries which communicate through the “proper” channel, rather than covertly. In this way, we also bypass existing impossibility results employing obfuscation [7]: We exclude by assumption adversaries which would use obfuscation.

Game 1 (Knowledge Extraction). The adversary \mathcal{A}_r wins the knowledge extraction game if and only if it outputs a series of objects in X_{pp} , for which the extractor \mathcal{X}_r fails to output the corresponding witness:

$$\text{let } \vec{x} \leftarrow \mathcal{A}_r(I, \text{aux}), \vec{w} \leftarrow \mathcal{X}_r(I, \text{aux}) \text{ in } \vec{x} \in X_{\text{pp}}^* \wedge \bigvee_{i=1}^{|\vec{x}|} (x_i, w_i) \notin \mathcal{R}_{\text{pp}}(I).$$

Crucial for composition are the existential quantifications, which combined state that we assume extraction *for all* of the following:

- Algorithms in $\text{Resp}_{\mathfrak{R}}$
- Input objects I
- Auxiliary inputs aux

This makes knowledge assumptions following Assumption 1 stronger than their typical property-based definitions. It is also non-standard as a result, as it relies on quantifiers *within* a probability experiment. While the adversarial win condition is well-defined, it is not necessarily computable. Nevertheless, quantifications are required for their usage in composable proofs.

2.1 Examples of Knowledge Assumptions

To motivate this definition, we demonstrate that it can be applied to various commonly used knowledge assumptions, including the knowledge of exponent assumption, the Algebraic Group Model and variants, and even to random oracles. We detail our flavour of the AGM here, and leave the details of the others to the full version of this paper [24, Appendix E]. Witnesses naturally seem to form a restricted expression language describing how to construct a knowledge-implying object. A more natural way to express the relation \mathcal{R} is often an evaluation function over witnesses, returning a knowledge-implying object.

The Algebraic Group Model. Assuming a distribution `groupSetup` providing a group \mathbb{G} and a generator g , we can recreate the Algebraic Group Model [17] as a knowledge assumption fitting Definition 1:

$$\begin{aligned} \mathfrak{R}_{\text{AGM}} &:= (\text{init}, X, W, \mathcal{R}) & W &:= \{ (\text{OP}, a, b) \mid a, b \in W \} \cup \\ \text{init} &:= \text{groupSetup} & & \{ (\text{INPUT}, i) \mid i \in X \} \cup \\ X &:= \mathbb{G} & & \{ \text{GENERATOR} \} \end{aligned}$$

$$\text{eval}(I, w) := \begin{cases} \text{eval}(g) \circ \text{eval}(h) & \text{if } w = (\text{OP}, g, h) \\ i & \text{if } w = (\text{INPUT}, i) \wedge i \in I \\ g & \text{if } w = \text{GENERATOR} \end{cases}$$

$$(x, w) \in \mathcal{R}(I) \iff x = \text{eval}(I, w)$$

3 Typed Networks of Random Systems

While it is not our goal to pioneer a new composable security framework, existing frameworks do not quite fit the needs of this paper. Notably, Universal Composability [8] has many moving parts, such as session IDs, control functions and different tapes which make the core issues harder to grasp. Constructive Cryptography [31] does not have a well-established notion of globality and makes variable numbers of interfaces difficult to implement, which make the transformations we will later perform trickier.

Furthermore, the analysis of knowledge assumptions benefits from a clear type system imposed on messages being passed – knowing which parts of messages encode objects of interest to knowledge assumptions (and which do not) makes the analysis more straightforward. Due to both of these reasons, we construct a compositional framework sharing many similarities with Constructive Cryptography, however using graphs (networks) of typed random systems as the basic unit instead of random systems themselves. Crucially, when we establish composition within this framework, we do so with respect to sets of valid distinguishers. This will allow us to permit only distinguishers which respect the knowledge assumption.

Our definitions can embed existing security proofs in Constructive Cryptography, and due to the close relation between composable frameworks, likely also those in other frameworks, such as UC. Notably, our results directly imply that primitives proven using knowledge assumptions under this framework can be directly used in place of hybrids in systems proven in Constructive Cryptography.

3.1 Type Definition

We introduce a rudimentary type system for messages passed through the network. For a casual reader, the details of this are unimportant – understanding that the type system allows filtering which parts of messages are relevant to a knowledge assumption and which aren't is sufficient to follow our construction.

Nevertheless, we formally define our type system as consisting of a unit type $\mathbb{1}$, empty type $\mathbb{0}$, sum and product types $\tau_1 + \tau_2 / \tau_1 \times \tau_2$, and the Kleene star τ^* . This type system was chosen to be minimal while still:

1. Allowing existing protocols to be fit within it. As most of cryptography operates on arbitrary length strings, $(\mathbb{1} + \mathbb{1})^*$, or finite mathematical objects, $\mathbb{1} + \dots + \mathbb{1}$, these can be embedded in the type system.
2. Allowing new types to be embedded in larger message spaces. The inclusion of sum types enables optional inclusion, while product types enables inclusion of multiple instances of a type alongside auxiliary information.

We stress that this type system may be (and will!) extended, and that a richer system may make sense in practice. Types follow the grammar

$$\tau \equiv \mathbb{0} \mid \mathbb{1} \mid \tau_1 + \tau_2 \mid \tau_1 \times \tau_2 \mid \tau^*,$$

and the corresponding expression language follows the grammar

$$E \equiv \top \mid \text{inj}_1(E) \mid \text{inj}_2(E) \mid (E_1, E_2) \mid \epsilon \mid E_1 :: E_2.$$

We will also use 2 to represent $\mathbb{1} + \mathbb{1}$, and 0 and 1 for $\text{inj}_1(\top)$ and $\text{inj}_2(\top)$ respectively. Formally, the typing rules are:

$$\begin{array}{c} \vdash \top : \mathbb{1} \\ \vdash x : \tau_1 \\ \hline \vdash \text{inj}_1(x) : \tau_1 + \tau_2 \\ \vdash x : \tau_2 \\ \hline \vdash \text{inj}_2(x) : \tau_1 + \tau_2 \\ \vdash x : \tau_1 \quad \vdash y : \tau_2 \\ \hline \vdash (x, y) : \tau_1 \times \tau_2 \\ \vdash \epsilon : \tau^* \\ \vdash x : \tau \quad \vdash \vec{x} : \tau^* \\ \hline \vdash x :: \vec{x} : \tau^* \end{array}$$

Note that there is no means to construct the empty type \emptyset .

Knowledge assumptions. We expand this basic type system by allowing objects to be annotated with a knowledge assumption. Specifically, given a knowledge assumption $\mathfrak{K} = (\text{init}, X, W, \mathcal{R})$, where init returns $\text{pp} : \tau$, and for all pp in the domain of init , both X_{pp} and W_{pp} are valid types, there are two additional types present:

1. The type of knowledge-implying objects in \mathfrak{K} : $[\mathfrak{K}_{\text{pp}}]$ (equivalent to X_{pp})
2. The type of witnessed objects in \mathfrak{K} with respect to an input set of knowledge I : $\forall I \subseteq X_{\text{pp}} : \langle \mathfrak{K}_{\text{pp}}, I \rangle$ (equivalent to $X_{\text{pp}} \times W_{\text{pp}}$)

Formally then, we define \mathfrak{K} types through the grammar

$$\tau \equiv \emptyset \mid \mathbb{1} \mid \tau_1 + \tau_2 \mid \tau_1 \times \tau_2 \mid \tau^* \mid [\mathfrak{K}_{\text{pp}}] \mid \langle \mathfrak{K}_{\text{pp}}, I \rangle,$$

with the corresponding expression grammar being

$$E \equiv \top \mid \text{inj}_1(E) \mid \text{inj}_2(E) \mid (E_1, E_2) \mid \epsilon \mid E_1 :: E_2 \mid [E]_{\mathfrak{K}_{\text{pp}}} \mid \langle E \rangle_{\mathfrak{K}_{\text{pp}}}^I.$$

Crucially, the types of messages may depend on prior interactions. This is particularly obvious with the set of input knowledge I , which will be defined as the set of all previously received $x : [\mathfrak{K}_{\text{pp}}]$, however it also applies to pp itself, which may be provided from another component of the system. This allows for the secure sampling of public parameters, or delegating this to a common reference string (CRS). The typing rules are extended with the following two rules, where X_{pp} and W_{pp} are type variable:

$$\frac{\vdash x : X_{\text{pp}} \quad \vdash w : W_{\text{pp}} \quad (x, w) \in \mathcal{R}_{\text{pp}}(I)}{\vdash \langle x, w \rangle_{\mathfrak{K}_{\text{pp}}}^I : \langle \mathfrak{K}_{\text{pp}}, I \rangle} \quad \frac{\vdash x : X_{\text{pp}}}{\vdash [x]_{\mathfrak{K}_{\text{pp}}} : [\mathfrak{K}_{\text{pp}}]}$$

3.2 Random Systems

We use the same basic building-block as Constructive Cryptography [31]: Random systems [32]. We briefly recap this notion:

Definition 2. An $(\mathcal{X}, \mathcal{Y})$ -random system \mathbf{F} is an infinite sequence of conditional probability distributions $P_{Y_i | X^i Y^{i-1}}^{\mathbf{F}}$ for $i \geq 1$, where X and Y distribute over \mathcal{X} and \mathcal{Y} respectively.

Specifically, random systems produce outputs in the domain \mathcal{Y} when given an input in \mathcal{X} , and are stateful – their behaviour can depend on prior inputs and outputs. [31] itself works with random systems based on an automaton with internal state; such an automaton can then also be constrained to a reasonable notion of feasibility, such as being limited to a polynomial number of execution steps with respect to some security parameter.

We will not go into depth on modelling computational security, as it is not the primary focus of this paper, however we will assume the existence of a feasibility notion of this type. We follow the approach of [29], and consider random systems as equivalence classes over probabilistic systems. We make a minor tweak to the setting of [31] as well, and use random-access machines instead of automata.

3.3 Typed Networks

We will consider networks of random systems (which can be considered as labelled graphs) as our basic object to define composition over.

Definition 3 (Cryptographic Networks). *A typed cryptographic network is a set of nodes N , satisfying the following conditions:*

1. Each node $n \in N$ is a tuple $n = (I_n, O_n, \tau_n, R_n, A_n)$ representing:
 - I_n a set of available input interfaces.
 - O_n a set of available output interfaces.
 - $\tau_n : I_n \cup O_n \rightarrow T$, a mapping from interfaces to their types.
 - R_n , a $(\sum_{i \in I_n} \tau_n(i), \sum_{o \in O_n} \tau_n(o))$ random system.
 - $A_n \subseteq I_n \cup O_n$, the subset of interfaces which behave adversarially.
2. Both input and output interfaces are unique within the network:

$$\forall a, b \in N : a \neq b \implies I_a \cap I_b = \emptyset \wedge O_a \cap O_b = \emptyset.$$

3. Matching input and output interfaces define directed channels in the implied network graph. Therefore, where $a, b \in N, i \in O_a \cap I_b$:
 - The interface types match: $\tau_a(i) = \tau_b(i)$.
 - The edges have a consistent adversarially: $i \in A_a \iff i \in A_b$.

We denote the set of all valid cryptographic networks by \mathfrak{N} .

This corresponds to a directed network graph whose vertices are nodes, and whose edges connect output interfaces to their corresponding input interface.

Composing multiple such networks is a straightforward operation, achieved through set union. While the resulting network is not necessarily valid, as it may lead to uniqueness of interfaces being violated, it can be used to construct any valid network out of its components. We also make use of a disjoint union, $A \uplus B$, by which we mean the union of A and B , while asserting that A and B are disjoint. Due to the frequency of its use, we will allow omitting the disjoint union operator, that is, we write AB to denote $A \uplus B$.

Definition 4 (Unbound Interfaces). *In a typed cryptographic network N , the sets of unbound input and output interfaces, written $I(N)$ and $O(N)$, respectively, are defined as the set of all tuples (i, τ) for which there exists $a \in N$ and $i \in I_a$ (resp. $i \in O_a$), where for all $b \in N$, $i \notin O_b$ (resp. $i \notin I_b$), with τ being defined as its type, $\tau_a(i)$. Furthermore, $IO_{\mathcal{H}}(N)$ is defined as the unbound honest interfaces: all $(i, \cdot) \in I(N) \cup O(N)$, where i is honest, that is, where $\forall a \in N : i \notin A_a$.*

We can define a straightforward token-passing execution mechanism over typed cryptographic networks, which demonstrates how each network behaves as a single random system.² We primarily operate with networks instead of reducing

² Termination is an issue here, in so far as the network may loop infinitely using message passing. We consider a non-terminating network to return the symbol \perp , although this might render the output uncomputable.

them to a single random system to preserve their structure: It allows easily applying knowledge assumptions to each part, and enables sharing components in parallel composition, a requirement for globality.

Definition 5 (Execution). *A typed cryptographic network N , together with an ordering of $I(N)$ and $O(N)$ defines a random system through token-passing execution, with the input and output domains $\sum_{(\cdot, \tau) \in I(N)} \tau / \sum_{(\cdot, \tau) \in O(N)} \tau$, respectively. Execution is defined through a stateful passing of messages – any input to N will be targeted to some $(i, \cdot) \in I(N)$. The input is provided to the random system R_a , for which $i \in I_a$. Its output will be associated with an $o \in O_a$. If there exists a $b \in N$ such that $o \in I_b$, it is forwarded to R_b , continuing in a loop until no such node exists. At this point, the output is associated with $(o, \cdot) \in O(N)$ (note that, if $O(N) = \emptyset$, the corresponding random system cannot be defined, as it has an empty output domain), and is encoded to the appropriate part of the output domain.*

The full version of this paper [24, Appendix A] goes into more detail on the semantics, formally describing the functions $\text{exec}(N, i, x)$ and $\text{execState}(N, i, x, \Sigma)$. In order to help with preventing interface clashes, we introduce a renaming operation.

Definition 6 (Renaming). *For a cryptographic network N , renaming interfaces a_1, \dots, a_n to b_1, \dots, b_n , is denoted by:*

$$N[a_1/b_1, \dots, a_n/b_n] := \{ m \in N \mid m[a_1/b_1, \dots, a_n/b_n] \}.$$

Where, for $m = (I_m, O_m, \tau_m, \cdot, A_m)$, $m[a_1/b_1, \dots, a_n/b_n]$ is defined by replacing each occurrence of a_i in the sets I_m , O_m and A_m with the corresponding b_i , as well as changing the domain of τ_m to accept b_i instead of a_i , with the same effect.

To ensure renaming does not introduce unexpected effects, we leave it undefined when any of the output names b_i are present in the network N , and are not themselves renamed (i.e. no a_j exists such that $a_j = b_i$). Likewise, we prohibit renaming where multiple output names are equal. For a set of cryptographic networks, the same notation denotes renaming on each of its elements.

When talking about valid distinguishers, these are sets of cryptographic networks closed under internal renaming.

Definition 7 (Distinguisher Set). *A set of distinguishers $\mathfrak{D} \subseteq \mathfrak{N}$ is any subset of \mathfrak{N} which is closed under internal renaming: For any $D \in \mathfrak{D}$, $\vec{n} = a_1/b_1, \dots, a_n/b_n$, where no a_i or b_i are in $I(D)$ or $O(D)$, $D[\vec{n}] \in \mathfrak{N} \implies D[\vec{n}] \in \mathfrak{D}$.*

Composition is also defined for distinguisher sets. Given a set of networks \mathfrak{D} and a network A , $\mathfrak{D}A$ is defined as the closure under internal renaming of $\{ DA \mid D \in \mathfrak{D} : DA \in \mathfrak{N} \}$. Observe that \mathfrak{N} is closed under composition, and therefore $\mathfrak{N}A \subseteq \mathfrak{N}$ for any $A \in \mathfrak{N}$. Renaming for distinguisher sets is defined similarly, allowing distinguisher sets to give special meaning to some *external* interfaces, but not to internal ones.

3.4 Observational Indistinguishability

Now that we have established the semantics of cryptographic networks, we can reason about their observational indistinguishability, defined through the statistical distances of their induced random systems combined with arbitrary distinguishers. The indistinguishability experiment is visualised in Fig. 1.

Definition 8 (Observational Indistinguishability). *Two cryptographic networks A and B are observationally indistinguishable with advantage ϵ with respect to the set of valid distinguishers \mathfrak{D} , written $A \stackrel{\epsilon, \mathfrak{D}}{\sim} B$, if and only if:*

- Their unbound inputs and outputs match: $I(A) = I(B) \wedge O(A) = O(B)$.
- For any network $D \in \mathfrak{D}$ for which DA and DB are both in \mathfrak{N} , with $I(DA) = I(DB) = (\cdot, \mathbb{1})$ and $O(DA) = O(DB) = (\cdot, \mathbb{2})$, the statistical distance $\delta^{\mathfrak{D}}(A, B)$ is at most ϵ , where

$$\delta^{\mathfrak{D}}(A, B) := \sup_{D \in \mathfrak{D}} \Delta^D(A, B)$$

$$\Delta^D(A, B) := |\Pr(DA = 1) - \Pr(DB = 1)|.$$

To simplify some corner cases, where $\forall D \in \mathfrak{D} : DA \notin \mathfrak{N} \vee DB \notin \mathfrak{N}$, we consider $\delta^{\mathfrak{D}}(A, B)$ to be 0 – in other words, we consider undefined behaviours indistinguishable.

The \mathfrak{D} term is omitted if it is clear from the context.

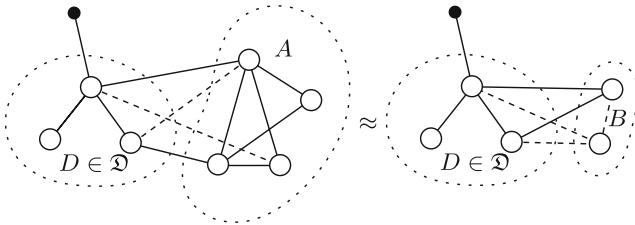


Fig. 1. A visual representation of a non-specific $A \stackrel{\mathfrak{D}}{\sim} B$ experiment, with solid lines representing honest interfaces, and dashed representing adversarial interfaces.

Observe that observational indistinguishability claims can be weakened:

$$A \stackrel{\epsilon, \mathfrak{D}_1}{\sim} B \wedge \mathfrak{D}_2 \subseteq \mathfrak{D}_1 \implies A \stackrel{\epsilon, \mathfrak{D}_2}{\sim} B \tag{1}$$

Lemma 1 (Observational Renaming). *Observational indistinguishability is closed under interface renaming:*

$$\forall A, B \in \mathfrak{N}, \mathfrak{D} \subseteq \mathfrak{N}, \epsilon, \vec{n} : A[\vec{n}], B[\vec{n}] \in \mathfrak{N} \wedge A \stackrel{\epsilon, \mathfrak{D}}{\sim} B \implies A[\vec{n}] \stackrel{\epsilon, \mathfrak{D}[\vec{n}]}{\sim} B[\vec{n}]$$

Proof. By precondition, we know that $I(A) = I(B) \wedge O(A) = O(B)$, that $\delta^{\mathfrak{D}}(A, B) \leq \epsilon$, and that \mathfrak{D} is closed under renaming. As renaming is restricted by definition to not create any new connections, $I(A[\vec{n}]) = I(A)[\vec{n}] = I(B)[\vec{n}] = I(B[\vec{n}])$, and likewise for O . As \mathfrak{D} remains unchanged, it remains to show that $\sup_{D \in \mathfrak{D}} |\Pr(DA[\vec{n}] = 1) - \Pr(DB[\vec{n}] = 1)| \leq \epsilon$.

Consider how, for $D \in \mathfrak{D}$, $(DA)[\vec{n}]$ and $(DB)[\vec{n}]$, are related to $D'(A[\vec{n}])$ and $D'(B[\vec{n}])$. If $(DA)[\vec{n}]$ is well-defined, then for $D' = D[\vec{n}]$, then $(DA)[\vec{n}] = D'(A[\vec{n}])$. Moreover, for any $D' \in \mathfrak{D}$, there exists some internal renaming \vec{m} such that $(D'[\vec{m}]A)[\vec{n}]$ and $(D'[\vec{m}]B)[\vec{n}]$ are well-defined, as the renaming \vec{m} can remove the potential name clashes introduced by \vec{n} . As \mathfrak{D} is closed under renaming, it is therefore sufficient to show that $\sup_{D \in \mathfrak{D}} |\Pr((DA)[\vec{n}] = 1) - \Pr((DB)[\vec{n}] = 1)| \leq \epsilon$. As the execution semantics of $(DA)[\vec{n}]$ and $(DB)[\vec{n}]$ does not use interface names, this is equivalent to $\sup_{D \in \mathfrak{D}} |\Pr(DA = 1) - \Pr(DB = 1)| = \delta^{\mathfrak{D}}(A, B) \leq \epsilon$. \square

Lemma 2 (Observational Equivalence). *Observational indistinguishability is an equivalence relation: It is **transitive**³ (Eq. 2), **reflexive** (Eq. 3), and **symmetric** (Eq. 4). For all $A, B, C \in \mathfrak{N}, \mathfrak{D} \subseteq \mathfrak{N}, \epsilon_1, \epsilon_2 \in \mathbb{R}$:*

$$A \stackrel{\epsilon_1, \mathfrak{D}}{\sim} B \wedge B \stackrel{\epsilon_2, \mathfrak{D}}{\sim} C \implies A \stackrel{\epsilon_1 + \epsilon_2, \mathfrak{D}}{\sim} C \quad (2)$$

$$A \stackrel{0, \mathfrak{D}}{\sim} A \quad (3)$$

$$A \stackrel{\epsilon_1, \mathfrak{D}}{\sim} B \iff B \stackrel{\epsilon_1, \mathfrak{D}}{\sim} A \quad (4)$$

Proof. We prove each part independently, given the well-known fact that statistical distance forms a pseudo-metric [31].

Transitivity. The equality of the input and output interfaces can be established by the transitivity of equality. The statistical distance is established through the triangle equality. Specifically, for all $D \in \mathfrak{D}$, $\Delta^D(A, C) \leq \Delta^D(A, B) + \Delta^D(B, C) \leq \epsilon_1 + \epsilon_2$. The only case where this is not immediate is if $DB \notin \mathfrak{N}$, which occurs in the case of an internal interface name collision – resolvable with renaming and use of Lemma 1. \square

Reflexivity. By the reflexivity of equality for input and output interfaces, and $\delta^{\mathfrak{D}}(A, A) = 0$ being established for pseudo-metrics. \square

Symmetry. By the symmetry of equality, and pseudo-metrics. \square

Lemma 3 (Observational Subgraph Substitution). *Observational indistinguishability is closed under subgraph substitution.*

$$\forall A, B, C \in \mathfrak{N}, \mathfrak{D} \subseteq \mathfrak{N}, \epsilon \in \mathbb{R} : A \stackrel{\epsilon, \mathfrak{D}^C}{\sim} B \iff CA \stackrel{\epsilon, \mathfrak{D}}{\sim} CB$$

³ Technically, due to the error terms, the relation is not transitive, but obeys a triangle inequality, and as a result it is also not an equivalence relation. We view this as a weak transitivity instead, as in practice, for negligible error terms, it behaves as such.

Proof. The equality of outgoing interfaces is trivially preserved under substitution, as the outgoing interfaces of A and B are the same by assumption.

We know that $\forall D \in \mathfrak{D}C : \Delta^D(A, B) \leq \epsilon$. Suppose there existed a distinguisher $D \in \mathfrak{D}$ such that $\Delta^D(CA, CB) \geq \epsilon$. Then, we can define $D' \in \mathfrak{D}C$ as DC , redrawing the boundary between distinguisher and network. By definition, $D' \in \mathfrak{D}C$, allowing us to conclude $\exists D' \in \mathfrak{D}C : \Delta^{D'}(A, B) \geq \epsilon$, arriving at a contradiction. The proof runs analogously in the opposite direction. \square

Corollary 1. For $\mathfrak{D} = \mathfrak{N}$, observational indistinguishability has the following, simpler statement for closure under subgraph substitution:

$$\forall A, B, C \in \mathfrak{N}, \epsilon : A \stackrel{\epsilon, \mathfrak{N}}{\sim} B \implies CA \stackrel{\epsilon, \mathfrak{N}}{\sim} CB$$

3.5 Composably Secure Construction of Networks

(Composable) simulation-based security proofs are then proofs that there exists an extension to one network connecting only on adversarial interfaces, such that it is observationally indistinguishable to another. We visualise and provide an example of construction in Fig. 2. Please be aware that despite the similar notation and model, the notion of construction described here differs significantly from that used in Constructive Cryptography.⁴

Definition 9 (Network Construction). A network $A \in \mathfrak{N}$ constructs another network $B \in \mathfrak{N}$ with respect to a distinguisher class \mathfrak{D} with simulator $\alpha \in \mathfrak{N}$ and error $\epsilon \in \mathbb{R}$, written $A \xrightarrow{\epsilon, \alpha, \mathfrak{D}} B$, if and only if $A \stackrel{\epsilon, \mathfrak{D}}{\sim} \alpha B$ and A and B have disjoint honest interfaces: $IO_{\mathcal{H}}(A) \cap IO_{\mathcal{H}}(B) = \emptyset$. The \mathfrak{D} term may be omitted when it is clear from the context, the α term may be omitted when it is of no interest, and the ϵ term may be omitted when it is negligible.

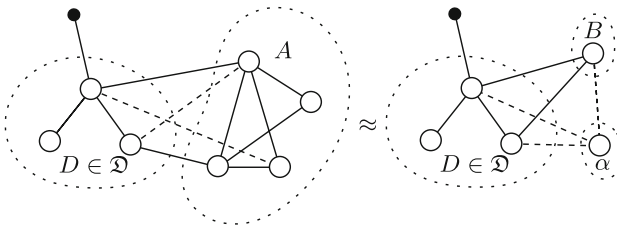


Fig. 2. A visual representation of a non-specific $A \xrightarrow{\alpha, \mathfrak{D}} B$ experiment.

⁴ Specifically, Constructive Cryptography’s construction moves the real-world protocol into the notation, becoming rather a statement of “resource A can be used to construct resource B ”. By contrast, this paper’s construction statement is closer to UC-emulation, being a statement of “system A is at least as secure as system B ”.

As with observational indistinguishability, network construction statements can be arbitrarily weakened. Furthermore, it is directly implied by indistinguishability:

$$A \succ_{\epsilon, \alpha, \mathfrak{D}_1} B \wedge \mathfrak{D}_2 \subseteq \mathfrak{D}_1 \implies A \succ_{\epsilon, \alpha, \mathfrak{D}_2} B \quad (5)$$

$$A \stackrel{\epsilon, \mathfrak{D}}{\sim} B \implies A \succ_{\epsilon, \emptyset, \mathfrak{D}} B \quad (6)$$

Theorem 1 (Generalised Composition). *Network construction is composable, in that it satisfies **transitivity** (Eq. 7), **subgraph substitutability** (Eq. 8), and **renameability** (Eq. 9). For all $A, B, C, \alpha, \beta \in \mathfrak{N}, \mathfrak{D} \subseteq \mathfrak{N}, \epsilon_1, \epsilon_2 \in \mathbb{R}, \vec{n}$:*

$$A \succ_{\epsilon_1, \alpha, \mathfrak{D}} B \wedge B \succ_{\epsilon_2, \beta, \mathfrak{D}\alpha} C \wedge \alpha\beta C \in \mathfrak{N} \implies A \succ_{\epsilon_1 + \epsilon_2, \alpha\beta, \mathfrak{D}} C \quad (7)$$

$$A \succ_{\epsilon_1, \alpha, \mathfrak{D}^C} B \wedge IO_{\mathcal{H}}(C) \cap IO_{\mathcal{H}}(\alpha B) = \emptyset \implies CA \succ_{\epsilon_1, \alpha, \mathfrak{D}} CB \quad (8)$$

$$A[\vec{n}], \alpha[\vec{n}]B[\vec{n}] \in \mathfrak{N} \wedge A \succ_{\epsilon_1, \alpha, \mathfrak{D}} B \implies A[\vec{n}] \succ_{\epsilon_1, \alpha[\vec{n}], \mathfrak{D}[\vec{n}]} B[\vec{n}] \quad (9)$$

Proof. We will prove each of the three properties separately.

Transitivity. By assumption, we know that $A \stackrel{\epsilon_1, \mathfrak{D}}{\sim} \alpha B$ and $B \stackrel{\epsilon_2, \mathfrak{D}}{\sim} \beta C$. By Lemma 3, we can conclude that $\alpha B \stackrel{\epsilon_2, \mathfrak{D}}{\sim} \alpha\beta C$. By transitivity (Lemma 2), we conclude that $A \stackrel{\epsilon_1 + \epsilon_2, \mathfrak{D}}{\sim} \alpha\beta C$.

Observe that β and C , as well as α and B have disjoint honest interfaces by assumption. As $B \stackrel{\epsilon_2, \mathfrak{D}}{\sim} \beta C$, they have the same public-facing interfaces. As $\alpha\beta C$ is well-defined, and as α and B have disjoint honest interfaces, so does α and βC . From each of α , β , and C 's honest interfaces being disjoint, we conclude that so are $\alpha\beta$ and C 's. \square

Closure under subgraph substitution. By assumption, we know $A \stackrel{\epsilon_1, \mathfrak{D}^C}{\sim} \alpha B$. By Lemma 3, we can conclude that $CA \stackrel{\epsilon_1, \mathfrak{D}}{\sim} C\alpha B$. As composition is a disjoint union, it is commutative, and therefore $C\alpha B = \alpha CB$. The interface disjointness requirement is satisfied by the precondition. \square

Closure under renaming. By assumption, we know $A \stackrel{\epsilon_1, \mathfrak{D}}{\sim} \alpha B$. By Lemma 1, we conclude that $A[\vec{n}] \stackrel{\epsilon_1, \mathfrak{D}[\vec{n}]}{\sim} (\alpha B)[\vec{n}] = \alpha[\vec{n}]B[\vec{n}]$. As $\alpha[\vec{n}]B[\vec{n}] \in \mathfrak{N}$, both $\alpha[\vec{n}]$ and $B[\vec{n}]$ are in \mathfrak{N} . As the honesty of edges remains unaffected by subgraph substitution, name collisions are not introduced, the disjointness requirement is also satisfied. Combined, this implies network construction in the renamed setting. \square

From the generalised composition theorem, which notably relies on modifying the distinguisher set (e.g. from \mathfrak{D} to $\mathfrak{D}\alpha$ in Eq. 7), we can infer operations similar to sequential and parallel composition in Constructive Cryptography, given $\mathfrak{D} = \mathfrak{N}$. For any \mathfrak{D} , identity also holds, due to the identity of indistinguishability, and indistinguishability lifting to construction.

Corollary 2 (Traditional Composition). For $\mathfrak{D} = \mathfrak{N}$, honest network construction has the following, simpler statements for **universal transitivity** (Eq. 10) and **universal closure under subgraph substitution** (Eq. 11). **Identity** (Eq. 12) holds regardless of \mathfrak{D} . For all $A, B, C, \alpha, \beta \in \mathfrak{N}, \epsilon_1, \epsilon_2 \in \mathbb{R}, \mathfrak{D} \subseteq \mathfrak{N}$:

$$A \succ_{\epsilon_1, \alpha, \mathfrak{N}} B \wedge B \succ_{\epsilon_2, \beta, \mathfrak{N}} C \wedge \alpha\beta C \in \mathfrak{N} \implies A \succ_{\epsilon_1 + \epsilon_2, \alpha\beta, \mathfrak{N}} C \tag{10}$$

$$A \succ_{\epsilon_1, \alpha, \mathfrak{N}} B \wedge I_{\mathcal{H}}(C) \cap I_{\mathcal{H}}(\alpha B) = \emptyset \implies CA \succ_{\epsilon_1, \alpha, \mathfrak{N}} CB \tag{11}$$

$$A \succ_{0, \emptyset, \mathfrak{D}} A \tag{12}$$

4 The Limited Composition of \mathfrak{K} -Networks

Having established a composition system which allows restricting the domain of permissible distinguishers, and having formalised the general notion of knowledge assumptions, we can now establish the main contribution of this paper: Permitting extraction from knowledge assumptions within a composable setting.

We use a similar idea to that of “algebraic adversaries” in the Algebraic Group Model [17], requiring random systems to output not only knowledge-implying objects, but also their corresponding witness. We then add new nodes to the network which gather all data extracted in this way in a central repository of knowledge for each knowledge assumption. Crucially, while the distinguisher supplies witnesses for all knowledge-implying objects it outputs, it is not capable of retrieving witnesses from other parts of the system.

Simulators are provided with read access to this repository, allowing the simulator to extract the knowledge it requires, but not any more about the behaviour of honest parties. The composition of constructions using knowledge assumptions is proven, provided the parts being composed do not both utilise the same knowledge assumption. In such a case, Theorem 1 provides a fall-back for what needs to be proven, namely that the simulator of one system does not permit distinguishing in the other system. At a technical level, modifications to Definition 3 are needed to allow types to depend on previously transmitted values. We note these formally in the full version of this paper [24, Appendix C]. This section serves as a detailed proof sketch, with [24, Appendix C] addressing some of the subtleties.

4.1 Knowledge Respecting Systems

The Algebraic Group Model [17] popularised the idea of “algebraic” adversaries, which must adhere to outputting group elements through a representation describing how they may be constructed from input group elements. Security proofs in the AGM assume that all adversaries are algebraic, and therefore the representation of group elements can be directly accessed in the reduction – by assumption it is provided by the adversary itself.

While this is equivalent to an extractor-based approach, for composition we will follow a similar “algebraic” approach. The premise is that for any random system R outputting (among other things) knowledge-implying objects in \mathfrak{K} , it is possible to construct an equivalent random system $\tilde{\mathfrak{K}}(R)$, which outputs the corresponding witnesses as well, provided each step of the random system is governed by a \mathfrak{K} -respecting algorithm.

Recall that a random system is an infinite sequence of probability distributions. As this is not in itself useful for applying Definition 1, we instead interpret them as an equivalence class over stateful, interactive, and probabilistic algorithms [29], with associated input and output types. For any such typed algorithm A and knowledge assumption \mathfrak{K}_{pp} , A can be separated into A_1 and A_2 , where A_1 outputs only a series of $[X_{pp}]$ values, and A_2 all the remaining information, such that A ’s output can be trivially reconstructed by inserting the $[X_{pp}]$ values of A_1 into the gaps in A_2 ’s outputs. Likewise, inputs can be split into the \vec{I} and aux inputs used in Game 1. Given this, we can define when a random system is \mathfrak{K} -respecting. Each such system has a corresponding “ \mathfrak{K} -lifted” system, which behaves “algebraically”, in that it also output witnesses.

Definition 10 (\mathfrak{K} -Respecting Systems). *A typed random system R is said to be \mathfrak{K} -respecting (or $R \in \text{RespSys}_{\mathfrak{K}}$), if and only if its equivalence class of stateful probabilistic algorithms contains a stateful algorithm A that when split as described in Subsect. 4.1 into A_1 and A_2 , satisfies $A_1 \in \text{Resp}_{\mathfrak{K}}$. For a set $\vec{\mathfrak{K}}$, $\text{RespSys}_{\vec{\mathfrak{K}}} := \bigcap_{\mathfrak{K} \in \vec{\mathfrak{K}}} \text{RespSys}_{\mathfrak{K}}$.*

Definition 11 ($\vec{\mathfrak{K}}$ -Lifted Systems). *A typed random system R induces a set of $\vec{\mathfrak{K}}$ -lifted random systems. This is defined by replacing, for any $\mathfrak{K} = (\cdot, X, W, \mathcal{R}) \in \vec{\mathfrak{K}}$, any (part of) an output from R with type $[\mathfrak{K}_{pp}]$ with (a part of) the output with type $\langle \mathfrak{K}_{pp}, I_{\mathfrak{K}_{pp}} \rangle$, where $I_{\mathfrak{K}_{pp}}$ is constructed as the set of all prior inputs to R of type $[\mathfrak{K}_{pp}]$. The output (part) $\langle x, w \rangle_{\mathfrak{K}_{pp}}^{I_{\mathfrak{K}_{pp}}}$ of the lifted system must be such that the equivalent output (part) on the unlifted system is $[x]_{\mathfrak{K}_{pp}}$, and $(x, w) \in \mathcal{R}_{\mathfrak{K}_{pp}}(I_{\mathfrak{K}_{pp}})$ with overwhelming probability.*

Theorem 2 ($\vec{\mathfrak{K}}$ -Lifting is Possible). *For random systems $R \in \text{RespSys}_{\vec{\mathfrak{K}}}$, at least one $\vec{\mathfrak{K}}$ -lifting of R , denoted $\vec{\mathfrak{K}}(R)$, exists.*

Proof. Split R into algorithms $A_{\mathfrak{K}}$ for each $\mathfrak{K} \in \vec{\mathfrak{K}}$, and A_* for the remaining computation, such that each $A_{\mathfrak{K}}$ outputs only $[\mathfrak{K}]$, and A_* outputs no such values, as described above. Then, by Assumption 1, there exist corresponding extractors $\mathcal{X}_{\mathfrak{K}}$ for each $\mathfrak{K} \in \vec{\mathfrak{K}}$, such that given the same inputs $\mathcal{X}_{\mathfrak{K}}$ outputs witnesses to the knowledge-implying objects output by $A_{\mathfrak{K}}$.

Replace $A_{\mathfrak{K}}$ with $A'_{\mathfrak{K}}$, which runs both $A_{\mathfrak{K}}$ and $\mathcal{X}_{\mathfrak{K}}$, and outputs $\langle x, w \rangle_{\mathfrak{K}}$, where $[x]_{\mathfrak{K}}$ is the output of $A_{\mathfrak{K}}$, and w is the output of $\mathcal{X}_{\mathfrak{K}}$. When reassembled into a random system, this modification satisfies Definition 11. \square

4.2 Lifting Networks for Knowledge Extraction

The set of $\vec{\mathfrak{R}}$ -respecting random systems $\text{RespSys}_{\vec{\mathfrak{R}}}$, along with the transformation $\vec{\mathfrak{R}}(R)$ for any $R \in \text{RespSys}_{\vec{\mathfrak{R}}}$, provides a means of lifting individual random systems. Applied to networks, it is clear something more is necessary – the lifting does not preserve the types of output interfaces, and to permit these to match again some additional changes need to be made to the networks. Looking forward, the lifted systems will interact with a separate, universal node REPO, which stores witnesses for the simulator to access.

We extend the notion of $\vec{\mathfrak{R}}$ -respecting to apply to networks, a network is $\vec{\mathfrak{R}}$ -respecting if and only if all vertices in it are also $\vec{\mathfrak{R}}$ -respecting (we will use $\text{RespNet}_{\vec{\mathfrak{R}}}$ as the corresponding set of $\vec{\mathfrak{R}}$ -respecting networks⁵). In lifting networks in this set, not only is each individual node lifted, but all outgoing connections are connected to a new node, which we name CHARON, which acts as a relay; re-erasing witnesses, while also informing a central repository of knowledge (outside of this network) of any witnesses it processes. We take the name from the ferryman of the dead in ancient Greek mythology, who in our case demands his toll in knowledge rather than coins. For any $\vec{\mathfrak{R}}$ -respecting network N , we define the lifting $\vec{\mathfrak{R}}(N)$ as follows:

Definition 12 (Network Lifting). *The network lifting $\vec{\mathfrak{R}}(N)$ for any cryptographic network $N \in \text{RespNet}_{\vec{\mathfrak{R}}}$ is defined to compose as expected. In particular, if there exists $\vec{\mathfrak{R}}', N' : N = \vec{\mathfrak{R}}'(N')$, then $\vec{\mathfrak{R}}(N)$ is defined as $(\vec{\mathfrak{R}} \cup \vec{\mathfrak{R}}')(N')$. Otherwise⁶, $\vec{\mathfrak{R}}(N)$ is defined as consisting of nodes n' for each node $n \in N$, where $R_{n'} = \vec{\mathfrak{R}}(R_n)$, and each output interface is renamed to a unique⁷ new interface name. For each output interface now named x , and previously named y in N , $\vec{\mathfrak{R}}(N)$ contains a new node $\text{CHARON}(\vec{\mathfrak{R}}, \text{adv})$, where adv denotes if the interface is adversarial, connected to free interfaces on the knowledge repository REPO and the public parameters for each knowledge assumption. Note that REPO is not part of the lifted network itself, which allows disjoint networks to remain disjoint when lifted.*

We specify the node CHARON in full detail in [24, Appendix B], along with the node REPO(\mathfrak{R}), which collects witnesses from CHARON, and provides adversarial access to them. REPO allows for some variation. For instance, it could

1. Return the set of all witnesses.
2. Return at most one witness.
3. Abort when no witness is available.

⁵ This set also forbids interface name clashes with REPO, ensuring this can be safely inserted, and is a subset of \mathfrak{N} .

⁶ Note that this is well-founded recursion, due to the base-case of $\vec{\mathfrak{R}} = \emptyset$, and as the order in which knowledge assumptions are added does not affect CHARON or REPO.

⁷ Where we assume uniqueness, this is assumed globally: In $\vec{\mathfrak{R}}(A)\vec{\mathfrak{R}}(B)$, the uniquely selected interface names should not clash, therefore being the same as $\vec{\mathfrak{R}}(AB)$.

4. For recursive witnesses (such as those used in the AGM and KEA assumptions), consolidate the witness into a maximal one, by recursively resolving (INPUT, i) terms.

We focus on **1**, as it is the simplest. The set of valid $\vec{\mathfrak{R}}$ -distinguishers $\mathfrak{D}_{\vec{\mathfrak{R}}}$ is defined with respect to REPO, where we assume the choice of variation is made separately for each knowledge assumption. Informally, it ensures that all parts of the distinguisher are $\vec{\mathfrak{R}}$ -lifted, and the distinguisher collects all witnesses in a central knowledge repository REPO, but does not retrieve witnesses from this, effectively only providing access to the simulator.

Definition 13 ($\vec{\mathfrak{R}}$ -Distinguishers). *The set of valid $\vec{\mathfrak{R}}$ -distinguishers $\mathfrak{D}_{\vec{\mathfrak{R}}}$, for any set of knowledge assumptions $\vec{\mathfrak{R}}$, is defined as the closure under internal renaming of*

$$\left\{ \vec{\mathfrak{R}}(N) \cup \bigcup_{\mathfrak{R} \in \vec{\mathfrak{R}}} \text{REPO}(\mathfrak{R}) \mid N \in \text{RespNet}_{\vec{\mathfrak{R}}} \right\}.$$

Note that as $N \in \text{RespNet}_{\vec{\mathfrak{R}}}$, it cannot directly connect to any of the REPO nodes.

As the number of REPO and public parameter interfaces may differ between the real and ideal world, we must normalise them before establishing indistinguishability. To do so, we wrap both worlds to contain an additional node, which we name \perp , which consumes all remaining interfaces, depending on the number already used. Formally, this is defined in [24, Appendix B.3].

Given these definitions, existing indistinguishability and construction results between $\vec{\mathfrak{R}}$ -respecting networks can be lifted to equivalent results between the lifted networks, with respect to $\vec{\mathfrak{R}}$ -distinguishers:

Lemma 4 (Indistinguishability Lifting). *If $A_1 A_2 \stackrel{\epsilon, \mathfrak{D}_{\vec{\mathfrak{R}}_1}}{\sim} B_1 B_2$, where for $i \in \{1, 2\}$, $A_i, B_i \in \text{RespNet}_{\vec{\mathfrak{R}}_2}$, $\vec{\mathfrak{R}}_1 \cap \vec{\mathfrak{R}}_2 = \emptyset$, and $\vec{\mathfrak{R}} := \vec{\mathfrak{R}}_1 \cup \vec{\mathfrak{R}}_2$, then:*

$$A_1 A_2 \stackrel{\epsilon, \mathfrak{D}_{\vec{\mathfrak{R}}_1}}{\sim} B_1 B_2 \implies A_1 \vec{\mathfrak{R}}_2(A_2) \stackrel{\epsilon, \mathfrak{D}_{\vec{\mathfrak{R}}}}{\sim} B_1 \vec{\mathfrak{R}}_2(B_2).$$

Lemma 5 (Construction Lifting). *For $A_{1,2}, B_{1,2}, \alpha_{1,2} \in \text{RespNet}_{\vec{\mathfrak{R}}_2}$ and $\vec{\mathfrak{R}}_1, \vec{\mathfrak{R}}_2$ where $\vec{\mathfrak{R}}_1 \cap \vec{\mathfrak{R}}_2 = \emptyset$, and $\vec{\mathfrak{R}} := \vec{\mathfrak{R}}_1 \cup \vec{\mathfrak{R}}_2$:*

$$A_1 A_2 \xrightarrow{\epsilon, \alpha_1 \alpha_2, \mathfrak{D}_{\vec{\mathfrak{R}}_1}} B_1 B_2 \implies A_1 \vec{\mathfrak{R}}_2(A_2) \xrightarrow{\epsilon, \alpha_1 \vec{\mathfrak{R}}_2(\alpha_2), \mathfrak{D}_{\vec{\mathfrak{R}}}} B_1 \vec{\mathfrak{R}}_2(B_2).$$

We visualise the construction experiment against a knowledge-respecting distinguisher set $\mathfrak{D}_{\vec{\mathfrak{R}}}$ in Fig. 3. This may be contrasted with Fig. 2, which does not have $\text{REPO}(\mathfrak{R})$, and does not allow the simulator to extract.

Lemma 6 ($\mathfrak{D}_{\vec{\mathfrak{R}}}$ Closure). *$\mathfrak{D}_{\vec{\mathfrak{R}}}$ is closed under sequential composition with lifted (with respect to $\vec{\mathfrak{R}}$) networks in $\text{RespNet}_{\vec{\mathfrak{R}}}$: $\forall R \in \text{RespNet}_{\vec{\mathfrak{R}}} : \mathfrak{D}_{\vec{\mathfrak{R}}} \vec{\mathfrak{R}}(R) \subseteq \mathfrak{D}_{\vec{\mathfrak{R}}}$*

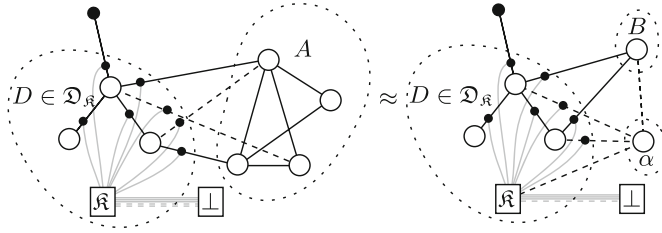


Fig. 3. A visual representation of a non-specific $A \xrightarrow{\alpha, \mathcal{D}_{\vec{\mathfrak{R}}}} B$ experiment. The small points denote $\text{CHARON}(\vec{\mathfrak{R}})$ nodes, while $\vec{\mathfrak{R}}$ denotes the $\text{REPO}(\vec{\mathfrak{R}})$ node. Public parameters have been omitted. Note that outside of D CHARON nodes are permitted, but not required.

Proof. Follows immediately from $\text{RespNet}_{\vec{\mathfrak{R}}}$ being closed under set union, and Definition 13 stating that any $\vec{\mathfrak{R}}$ -lifted network has a corresponding distinguisher in $\mathcal{D}_{\vec{\mathfrak{R}}}$. \square

As a stricter set of knowledge assumptions corresponds to a smaller set of permissible distinguishers, indistinguishability and construction results can be transferred to larger sets of knowledge assumptions. A proof without knowledge assumptions is clearly ideal – it still holds, regardless which knowledge assumptions are added.

Lemma 7 (Knowledge Weakening). *In addition to weakening with respect to a subset of distinguishers being possible, weakening is also possible for distinguishers with a greater set of knowledge assumptions. For all $A, B, C, \alpha \in \mathfrak{N}, \vec{\mathfrak{R}}_1, \vec{\mathfrak{R}}_2$, where $\vec{\mathfrak{R}}_1 \subseteq \vec{\mathfrak{R}}_2$:*

$$A \stackrel{\epsilon, \mathcal{D}_{\vec{\mathfrak{R}}_1}}{\sim} B \implies A \stackrel{\epsilon, \mathcal{D}_{\vec{\mathfrak{R}}_2}}{\sim} B \tag{13}$$

$$A \xrightarrow{\epsilon, \alpha, \mathcal{D}_{\vec{\mathfrak{R}}_1}} B \implies A \xrightarrow{\epsilon, \alpha, \mathcal{D}_{\vec{\mathfrak{R}}_2}} B \tag{14}$$

4.3 A Restricted Composition Theorem

The rules established in Theorem 1 still hold, and it is clear why a simplification as in Corollary 2 is not possible – it assumes that the distinguisher set \mathcal{D} is closed under sequential composition with simulators and networks, which is not the case for $\mathcal{D}_{\vec{\mathfrak{R}}}$.

Theorem 1 already provides a sufficient condition for what needs to be proven to enable this composition, however we can go a step further: While $\mathcal{D}_{\vec{\mathfrak{R}}}$ is not closed under sequential composition with arbitrary networks, it *is* closed under sequential composition with knowledge-lifted networks. We can use this fact to establish a simplified composition theorem when composing with a $\vec{\mathfrak{R}}$ -lifted proof or network component. We observe that this implies composition with proofs which do not utilise knowledge assumptions, as they are isomorphic to $\vec{\mathfrak{R}} = \emptyset$.

In particular, Constructive Cryptography proofs directly imply construction in the context of this paper as well, and can therefore be composed with protocols utilising our framework freely.

Theorem 3 (Knowledge Composition). *When composing proofs against $\vec{\mathfrak{K}}_1$ or $\vec{\mathfrak{K}}_2$ distinguishers, where $\vec{\mathfrak{K}}_1 \cap \vec{\mathfrak{K}}_2 = \emptyset$, and $\vec{\mathfrak{K}} := \vec{\mathfrak{K}}_1 \cup \vec{\mathfrak{K}}_2$, the following simplified composition rules of **transitivity** (Eq. 15) and **subgraph substitution** (Eq. 16) apply. For all $A, B, \alpha \in \text{RespNet}_{\vec{\mathfrak{K}}_2}, F \in \text{RespNet}_{\vec{\mathfrak{K}}}, C, D, E, \beta, \gamma \in \mathfrak{N}, \epsilon, \epsilon_1, \epsilon_2$.*

$$\left[\begin{array}{c} A \xrightarrow{\epsilon_1, \alpha, \mathcal{D}_{\vec{\mathfrak{K}}_1}} B \\ \wedge \\ B \xrightarrow{\epsilon_2, \beta, \mathcal{D}_{\vec{\mathfrak{K}}_2}} C \end{array} \right] \wedge \alpha\beta C \in \mathfrak{N} \implies A \xrightarrow{\epsilon_1 + \epsilon_2, \vec{\mathfrak{K}}_2(\alpha)\beta, \mathcal{D}_{\vec{\mathfrak{K}}}} C \quad (15)$$

$$D \xrightarrow{\epsilon, \gamma, \mathcal{D}_{\vec{\mathfrak{K}}}} E \wedge IO_{\mathcal{H}}(F) \cap IO_{\mathcal{H}}(\gamma E) = \emptyset \implies \vec{\mathfrak{K}}(F)D \xrightarrow{\epsilon, \gamma, \mathcal{D}_{\vec{\mathfrak{K}}}} \vec{\mathfrak{K}}(F)E \quad (16)$$

4.4 Reusing Knowledge Assumptions

Theorem 3 and its supporting lemmas prominently require disjoint sets of knowledge assumptions. The primary reason for this lies in the definition of $\vec{\mathfrak{K}}$ using the union of the knowledge assumptions $\vec{\mathfrak{K}}_1$ and $\vec{\mathfrak{K}}_2$ – all statements could also be made using a disjoint union here instead. If knowledge assumptions were not disjoint, this would place an unreasonable constraint on the distinguisher however: It would prevent it from copying information from one instance of a knowledge assumption to another instance of the same knowledge assumption, something any adversary is clearly capable of doing.

Equality for knowledge assumptions is not really well defined, and indeed knowledge assumptions may be related. The disjointness requirement is therefore more a statement of intent than an actual constraint, and we stress the importance of it for reasonably constraining the distinguisher set here: If the distinguisher is constrained with respect to two instances of knowledge assumptions which are related, it may not be permitted to copy from one two to another for instance, an artificial and unreasonable constraint.

Care must be taken that knowledge stemming from one knowledge assumption does not give an advantage in another. In many – but not all – cases this is easy to establish, for instance, we conjecture that multiple instances with the AGM with independently sampled groups are sufficiently independent. If this care is not taken, the union of two knowledge assumptions may be greater than the sum of its parts, as using both together prevents the distinguisher from exploiting structural relationships between the two, something a real adversary may do.

5 zk-SNARKs with an Updateable Reference String

To demonstrate the usefulness of this framework, we will showcase an example of how it can lift existing results to composability. For brevity, we sketch the approach instead of providing it in full detail. Specifically, we sketch how Groth’s zk-SNARK [20], due to being simulation extractable [1], can be used to construct an ideal NIZK. Our methodology applies to any SNARK scheme which permits proof simulation and extraction through the AGM. Further, we sketch how, when used for a SNARK requiring an *updateable* reference string, a round-robin protocol to produce the reference string can be used to instantiate the NIZK from only the CRS providing the AGM parameters.

Our approach for NIZKs is similar to $C\emptyset C\emptyset$ [28], with the difference that no additional transformation is necessary to extract witnesses, as these are provided through $\mathfrak{K}_{\text{bAGM}}$ -lifting and the simulator’s ability to extract from the knowledge assumption. The round-robin update follows [25] for its composable treatment updateable reference strings, simplified to a setting with fixed participants.

Once our proof sketch is complete, we also give a clear example of why universal composition is not possible with knowledge assumptions: Specifically, we construct a complementary ideal network and simulator which clearly violates the zero-knowledge properties of the NIZK, and allows distinguishing the real and ideal worlds. We stress that this is only possible due to it extracting from the same knowledge assumption.

5.1 Construction

Our construction is in two parts, each consisting of a real and ideal world. We describe and illustrate the set-up and behaviour here, leaving a more formal description of the exact behaviour to the full version of this paper [24, Appendix D]. Throughout the construction, we assume a set of n parties, identified by an element in \mathbb{Z}_n . We assume static corruption with at least one honest party – specifically we assume a set of adversaries $\mathcal{A} \subset \mathbb{Z}_n$, and a corresponding set of honest parties $\mathcal{H} := \mathbb{Z}_n \setminus \mathcal{A}$. These sets cannot be used in the protocols themselves, but are known to the distinguisher and non-protocol nodes (that is, they can be used to define ideal behaviour).

SNARKs. The highest level ideal world consists of a proof-malleable NIZK node (NIZK, see [24, Appendix D.2]), following the design of $C\emptyset C\emptyset$ [28]. In the corresponding real-world, we use a zk-SNARK scheme $\mathcal{S} = (S, T, P, \text{Prove}, \text{Verify}, \text{SimProve}, \mathcal{X}_w)$ satisfying the standard properties of correctness, soundness, and zero-knowledge in the random oracle model with SRS. Here S , T , and P , are the structure function, trapdoor domain, and permissible permutations⁸ of the

⁸ This can also capture non-updateable reference strings, when parameterised with the set of permissible permutations $P = \{\text{id}\}$. Notably this allows us to capture Groth’s zk-SNARK, while not excluding updateable zk-SNARKs such as Plonk [18] and Sonic [30].

structured reference strings, as given in [25]. *SimProve* should take as inputs only the statement x and trapdoor $\tau \in T$. In addition, \mathcal{S} should be simulation extractable with respect to the AGM – after any arbitrary interaction, \mathcal{X}_w should be able to produce the witness for any valid statement/proof pair, with the sole exception that the proof was generated with *SimProve*. Such white-box simulation extractability has been under-studied for zk-SNARKs, although it has been established for Groth’s zk-SNARK [1], and is plausible to hold in the AGM for most SNARKs. For this reason, we rely on Groth’s zk-SNARK to concretely instantiate this example, although we conjecture it applies to other SNARKs – and indeed part of the result *can* only apply to other SNARKs.

In the real world an adversarially biased (updateable) structured reference string (SRS, see [24, Appendix D.3]), parameterised for the SNARK’s reference string, is available. Further, for each honest party $j \in \mathcal{H}$, an instance of the SNARK protocol node (SNARK-NODE(j), see [24, Appendix D.3]) is available, which connects to the corresponding party’s SRS interface, and runs the SNARK’s Prove and Verify algorithms when queried. In both worlds, the $\mathfrak{R}_{\text{bAGM}}$ public parameters are provided by a node \mathbb{G} (see [24, Appendix D.1]). Finally, the SNARK’s Prove and Verify algorithms make use of a random oracle, which is available in the real world, providing query interfaces to all parties (we do not treat the random oracle as a knowledge assumption in this example).

The ideal-world therefore consists of $\{\text{NIZK}, \mathbb{G}\}$ (and the simulator, which will be introduced in the security analysis), and the real-world consists of $\text{SNARK} \uplus \{\text{SRS}, \text{RO}, \mathbb{G}\}$, where $\text{SNARK} := \{ \text{SNARK-NODE}(j) \mid j \in \mathcal{H} \}$. The topology of both worlds is sketched in Fig. 4.

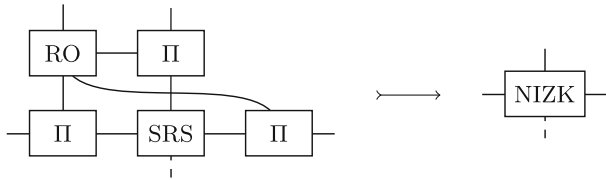


Fig. 4. The SNARK to NIZK topologies. SNARK-NODE is represented by Π , and the public parameter node \mathbb{G} is omitted for clarity.

Round-robin SRS. If the reference string used in the SNARK scheme \mathcal{S} is also updateable in the sense of [25], we can construct the SRS itself through a round-robin update protocol. We assume therefore that \mathcal{S} is additionally parameterised by algorithms *ProveUpd* and *VerifyUpd* allowing the proving and verification of update proofs, the permutation lifting \dagger which maps permutations in P to permutations over the structure, and the algorithms \mathcal{S}_p and \mathcal{X}_p used by the simulator to simulate update proofs and extract permutations from updates respectively. A notable difference again with respect to the extraction is that

it should be with respect to the AGM, rather than with respect to a NIZK as presented in [25].

The ideal world in this part matches part of the SNARK real world previously, consisting of the pair of nodes $\{\text{SRS}, \mathbb{G}\}$. The real-world consists of a node providing synchronous, authenticated broadcast (BCAST, see [24, Appendix D.4]), and for each honest party $j \in \mathcal{H}$, a round-robin protocol node (RR-NODE(j), see [24, Appendix D.4]).

The SRS node requires each honest party to request initialisation, which in the round-robin node is mapped to a) reconstructing the current SRS, and b) broadcasting a randomly sampled update to it. As the real-world has no means of identifying honest parties, it requires *all* parties to broadcast a valid update before the reference string can be used. The adversary has access to the broadcast directly for corrupted parties to produce these updates.

The ideal-world therefore consists of $\{\text{SRS}, \mathbb{G}\}$ (and simulator), and the real-world consists of $\text{RR-SETUP} \uplus \{\text{BCAST}, \mathbb{G}\}$, where $\text{RR-SETUP} := \{\text{RR-NODE}(j) \mid j \in \mathcal{H}\}$. The topology of both worlds is sketched in Fig. 5.

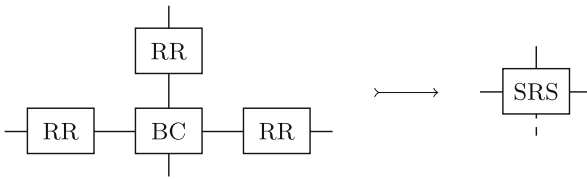


Fig. 5. The round-robin setup to SRS topologies. BCAST and RR-NODE are abbreviated to BC and RR respectively, and the public parameter node \mathbb{G} is omitted for clarity.

5.2 Security Analysis

This example is interesting for two reasons: Firstly, it provides a concrete way to realise a composable NIZK, and secondly it showcases (when the second optional stage of realising the SRS is used) special-case composition between two constructions using the same knowledge assumption, and what this requires of the corresponding simulators, as both simulators extract from $\text{REPO}(\mathcal{R}_{\text{bAGM}})$.

The two simulators, α for the simulator between SNARK and NIZK, and β for the simulator between RR-SETUP and SRS, are specified in full detail in the full version of this paper [24, Appendix D.5], although we sketch the most important aspects here. Notably, α needs to extract the witnesses from adversarial SNARK proofs, and β needs to extract the underlying trapdoor permutation from adversarial updates.

Round-robin SRS. The simulator β for the round-robin SRS setup emulates the broadcast node BCAST towards the adversary, and when notified of an honest party’s initialisation, does one of two things: For the first honest party, it queries

the honest SRS part from SRS, and simulates the corresponding update proof using the simulator \mathcal{S}_ρ , as it knows the full trapdoor to use for this. For subsequent honest updates, it simply simulates the update protocol. In either case, the update is internally recorded to emulate the corresponding broadcast.

When an adversarial broadcast is received, the update is verified against the current SRS. If it succeeds, it is updated, and the corresponding permutation is extracted from the update proof (using $\mathfrak{R}_{\text{bAGM}}$), and recorded. Specifically, the extractor \mathcal{X}_p , given oracle access to $\text{REPO}(\mathfrak{R}_{\text{bAGM}})$, extracts the permutation from any update proof ρ . Observe that a) such a permutation exists by the nature of the verification of update proofs, and b) the only group elements which the simulator *cannot* extract from are those in the honest SRS component produced by the SRS node.

Given this, the adversary cannot create a valid update for which the permutation is not extractable, unless it reuses (part of) the honest update. This would directly require inverting its structure before re-applying (part of) it again however, or the adversary extracting the permutation itself. In either case, this amounts to breaking a discrete logarithm for SNARKs we considered, which we assume computationally infeasible.

Theorem 4 (Round-Robin uSRS). *Given the computational hardness of the structure function S , as well as computational hardness to extract a trapdoor permutation p from an update proof ρ :*

$$\mathfrak{R}(\text{RR-SETUP}) \uplus \{\mathbb{G}\} \xrightarrow{\beta, \mathfrak{D}_{\mathfrak{R}}} \{\text{SRS}, \mathbb{G}\}$$

Proof (sketch). The simulated broadcast network the adversary has access to behaves identically between the real protocol and the simulated one, due to identical execution, except for the first honest update. This is distributed uniformly randomly in the space of possible permutations in both cases.

As the simulator reproduces a permutation which applies precisely all updates after the first honest one, and the first honest update is distributed the same in both worlds, the permutation the simulator applies to the honest trapdoor causes it to be distributed as in the real protocol. Further, both worlds abort if and only if the reference string is queried prior to full initialisation in both worlds. By the reasoning above and the hardness assumptions, extraction of adversarial updates always succeeds, and as a result the simulated update proof also succeeds. \square

SNARK. The SNARK simulator α both faithfully simulates the SRS node, creates simulated proofs for honest proving queries, and extracts witnesses using \mathcal{X}_w (which is given access to $\text{REPO}(\mathfrak{R}_{\text{bAGM}})$) from adversarial proofs when requested by the NIZK node. Finally, if the simulator fails to extract a witness when asked for one for a valid proof, it requests a maul. The SNARK simulator can co-exist with the SRS simulator provided above, provided that the SRS update proofs cannot be interpreted as NIZK proofs (with the trapdoor permutation as a witness) themselves, or transformed into ones. In practice, this is not the case, as the

AGM allows only for very specific transformations of group elements, and mapping update proofs to a corresponding NIZK would involve first solving DLOG before re-encoding the witness as a polynomial in most SNARKs.

Theorem 5 (SNARK Protocols Construct NIZKs). *For any secure SNARK scheme \mathcal{S} :*

$$\mathfrak{R}(\text{SNARK}) \uplus \{\text{SRS}, \mathfrak{R}(\text{RO}), \mathbb{G}\} \xrightarrow{\alpha, \mathcal{D}_{\mathfrak{R}}} \{\text{NIZK}, \mathbb{G}\}. \tag{17}$$

Additionally, if \mathcal{S} is updateable (and therefore β is well-defined), and update proofs cannot be transformed into NIZK proofs with the trapdoor permutation as a witness:

$$\mathfrak{R}(\text{SNARK}) \uplus \{\text{SRS}, \mathfrak{R}(\text{RO}), \mathbb{G}\} \xrightarrow{\alpha, \mathcal{D}_{\mathfrak{R}\beta}} \{\text{NIZK}, \mathbb{G}\}. \tag{18}$$

Proof (sketch). All honestly generated proofs will verify in both worlds, by definition in the ideal world, and by the correctness of the SNARK in the real world. Further, the proofs themselves are indistinguishable, by the zero-knowledge property of the SNARK.

Adversarial proofs which fail to verify will also be rejected in the ideal world, as the simulator will refuse to provide a witness, causing them to be rejected. As per the above, the extractor \mathcal{X}_w is able to (using $\text{REPO}(\mathfrak{R}_{\text{bAGM}})$) extract the witnesses for any adversarial proof which *does* verify, except for cases of malleability. As \mathcal{S} is only (at most) proof-malleable, the simulator can, and does, account for this by attempting to create a mauled proof when extraction fails.

The simulator provides the ideal-world simulation of the SRS node, which is emulated faithfully except that the simulator has access to the trapdoor. As a result, this part of the system cannot be used to distinguish. We conclude that Eq. 17 holds.

For Eq. 18, it remains to be shown that α and β do not interfere: In particular, that neither prevents the other from extracting where they need to, and that neither reveal information due to their extractions which would provide the distinguisher a non-negligible advantage. As β only interacts with the SRS, and this is not changed once all users have submitted their contribution, and α requires the SRS to be fully initialised before it is used, α will not prevent β from extracting – it does nothing while β is run.

Knowledge of the group elements exchanged during the update phase also does not assist the distinguisher in constructing a witness for any statement, as it can simulate them locally by running the honest update process. Therefore α can still fully extract in all cases.

Finally, due to the strict temporal order, β can, by definition, not assist the distinguisher in extracting any additional differences between SNARK and α (and the SRS part is emulated faithfully, preventing it there). Likewise, α cannot assist the distinguisher in extracting anything meaningful from β , as this would imply a NIZK witness containing the permutation of an honest update. As these are sampled locally, and the corresponding update proofs cannot be transformed into NIZK proofs, α cannot be leveraged to extract them. \square

Corollary 3. *For an updateable SNARK scheme \mathcal{S} whose update proofs cannot be transformed into NIZK proofs with the trapdoor permutation as a witness:*

$$\mathfrak{R}(\text{SNARK} \uplus \text{RR-SETUP}) \uplus \{\mathfrak{R}(\text{RO}), \mathbb{G}\} \xrightarrow{\beta\alpha, \mathcal{D}_{\vec{\mathfrak{R}}}} \{\text{NIZK}, \mathbb{G}\}$$

The topology for the composed statement arises naturally from the topologies of its parts, as shown in Fig. 6.

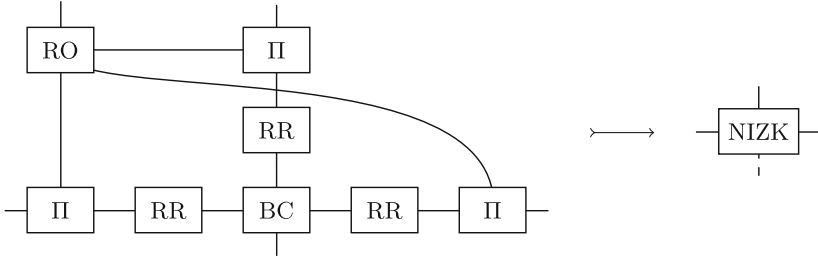


Fig. 6. The combined full example topology, arising from the composition of prior components, again with \mathbb{G} omitted for clarity.

Proof. From Theorem 4, Theorem 3 and Eq. 16, we can conclude that $\mathfrak{R}(\text{SNARK} \uplus \text{RR-SETUP}) \uplus \{\mathfrak{R}(\text{RO}), \mathbb{G}\} \xrightarrow{\beta, \mathcal{D}_{\vec{\mathfrak{R}}}} \mathfrak{R}(\text{SNARK}) \uplus \{\text{SRS}, \mathfrak{R}(\text{RO}), \mathbb{G}\}$. The corollary then follows from Theorem 5, Theorem 3, Eq. 18 and Eq. 15. \square

5.3 The Impossibility of General Composition

The two parts of Theorem 3 are limited when compared to Corollary 2 in two separate, but related ways: The closure under subgraph substitution requires the added node to be a $\vec{\mathfrak{R}}$ -wrapped node, and transitivity requires the two composing proofs to use separate knowledge assumptions.

We will demonstrate that the nicer results from Corollary 2 are not achievable with respect to knowledge-respecting distinguishers, by means of a small counterexample for both situations.

Theorem 6 (Subgraph Substitution is Limited). *Subgraph substitution with knowledge assumptions does not universally preserve secure construction.* $\exists A, B, C, \alpha \in \mathfrak{N}, \epsilon \in \mathbb{R}, \vec{\mathfrak{R}}:$

$$A \xrightarrow{\epsilon, \alpha, \mathcal{D}_{\vec{\mathfrak{R}}}} B \not\Rightarrow CA \xrightarrow{\epsilon, \alpha, \mathcal{D}_{\vec{\mathfrak{R}}}} CB$$

Proof (sketch). Let A be the Groth-16 real world, and B be the NIZK ideal world respectively, with α being their simulator, and $\vec{\mathfrak{R}}$ being $\{\mathfrak{R}_{\text{bAGM}}\}$. Let C be a node which receives elements in X_{pp} , queries $\text{REPO}(\mathfrak{R}_{\text{bAGM}})$, and returns the witness to the distinguisher.

Then the following distinguisher can trivially distinguish the two worlds: a) Make any honest proving query. b) Request extraction. c) Output whether or not extraction succeeded. \square

Theorem 7 (Transitivity is Limited). *Construction with knowledge assumptions is not universally transitive. $\exists A, B, C, \alpha, \beta \in \mathfrak{N}, \epsilon_1, \epsilon_2 \in \mathbb{R}, \mathfrak{D}_{\vec{\mathfrak{R}}}$:*

$$A \xrightarrow{\epsilon_1, \alpha, \mathfrak{D}_{\vec{\mathfrak{R}}}} B \wedge B \xrightarrow{\epsilon_2, \beta, \mathfrak{D}_{\vec{\mathfrak{R}}}} C \not\Rightarrow A \xrightarrow{\epsilon_1 + \epsilon_2, \alpha\beta, \mathfrak{D}_{\vec{\mathfrak{R}}}} C$$

Proof (sketch). Let B be the Groth-16 real world, and C be the NIZK ideal world respectively, with β being their simulator, and $\vec{\mathfrak{R}}$ being $\{\mathfrak{R}_{\text{bAGM}}\}$. Let A be Groth-16 with additional interfaces for each party to reveal any witnesses of broadcast proofs, which are shared through an additional broadcast channel. Let α reproduce this functionality by extracting witnesses from the provided proofs.

Then a distinguisher which makes an honest proof and extracts it will receive the witness in the real and hybrid world, but not in the ideal world, where the knowledge extraction of the proof will fail, as it is simulated by β . It is therefore possible to distinguish, and transitivity does not hold. \square

6 Conclusion

In this paper, we have for the first time demonstrated the composability of a white-box extractable zk-SNARK, without any transformations or modifications applied, and not compromising on succinctness. This result has immediate applications in the many systems which use zk-SNARKs and non-interactive zero-knowledge, reducing the gap between the theory and practice of composable systems relying on SNARKs. Our results are sufficiently general to hope for similar benefits when applied to other primitives utilising knowledge assumptions.

We nonetheless leave a number of pressing issues to future work: In many cases knowledge assumptions *are* reused. For instance many different protocols rely on the same groups, with the BLS12-381 and BN-254 curves being de-facto standards for zk-SNARK computation due to their direct use in major software implementations [3, 5]. To what degree this reuse is harmful, if at all, is a question of immediate interest and concern. This is compounded by a recent interest in *recursive* zk-SNARKs, such as Halo [6] – a natural compositional definition of which would construct a zk-SNARK from itself repeatedly. We hope that this work paves the way for a proper compositional treatment of such recursive constructions.

The foundations of knowledge assumptions also require further fleshing out to match reality more fully. It is clear that some knowledge assumptions are related, for instance the knowledge of exponent assumption is implied by the AGM. More interestingly, non-interactive zero-knowledge can itself be seen as a knowledge assumption – knowledge of a valid proof implying knowledge of the witness. Exploring the formal relationship between different knowledge assumptions and

expanding the model to fit these (for instance, by permitting public parameters to be adversarially influenced) may give valuable insight into the nature of these assumptions.

Acknowledgements. The second and third author were partially supported by the EU Horizon 2020 project PRIVILEGE #780477.

References


1. Baghery, K., Kohlweiss, M., Siim, J., Volkhov, M.: Another look at extraction and randomization of Groth’s zk-SNARK. In: FC 2021, March 2021
2. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474. IEEE Computer Society Press, May 2014
3. Ben-Sasson, E., et al.: libsnark: a C++ library for zkSNARK proofs (2017). <https://github.com/scipr-lab/libsnark>
4. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, A. (ed.) ITCS 2012, pp. 326–349. ACM, January 2012
5. Bowe, S.: bellman (2018). <https://github.com/zkcrypto/bellman>
6. Bowe, S., Grigg, J., Hopwood, D.: Halo: recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019). <https://eprint.iacr.org/2019/1021>
7. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 236–261. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_10
8. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
9. Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 449–460. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_37
10. Canetti, R., Dakdouk, R.R.: Towards a theory of extractable functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 595–613. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_35
11. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_2
12. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_26
13. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_36
14. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12

15. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_10
16. Ethereum Foundation: ZK-Rollups. <https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-rollups/>
17. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
18. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019). <https://eprint.iacr.org/2019/953>
19. Gazi, P., Kiayias, A., Zindros, D.: Proof-of-stake sidechains. In: 2019 IEEE Symposium on Security and Privacy, pp. 139–156. IEEE Computer Society Press, May 2019
20. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
21. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_24
22. Groth, J., Maller, M.: Snarky signatures: minimal signatures of knowledge from simulation-extractable SNARKs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 581–612. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_20
23. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055744>
24. Kerber, T., Kiayias, A., Kohlweiss, M.: Composition with knowledge assumptions. Cryptology ePrint Archive, Report 2021/165 (2021). <https://eprint.iacr.org/2021/165>
25. Kerber, T., Kiayias, A., Kohlweiss, M.: Mining for privacy: how to bootstrap a snarky blockchain. In: FC 2021, March 2021
26. Kerber, T., Kiayias, A., Kohlweiss, M., Zikas, V.: Ouroboros cryptsinous: privacy-preserving proof-of-stake. In: 2019 IEEE Symposium on Security and Privacy, pp. 157–174. IEEE Computer Society Press, May 2019
27. Kiayias, A., Liu, F.H., Tselekounis, Y.: Practical non-malleable codes from l-more extractable hash functions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1317–1328. ACM Press, October 2016
28. Kosba, A., et al.: $C\mathcal{C}\mathcal{O}$: a framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093 (2015). <https://eprint.iacr.org/2015/1093>
29. Lanzenberger, D., Maurer, U.: Coupling of random systems. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 207–240. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64381-2_8
30. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019, pp. 2111–2128. ACM Press, November 2019

31. Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) TOSCA 2011. LNCS, vol. 6993, pp. 33–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27375-9_3
32. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_8
33. 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, May 2013
34. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18



Non-interactive Batch Arguments for NP from Standard Assumptions

Arka Rai Choudhuri^(✉) , Abhishek Jain, and Zhengzhong Jin

Johns Hopkins University, Baltimore, USA
{achoud, abhishek, zzjin}@cs.jhu.edu

Abstract. We study the problem of designing *non-interactive batch arguments* for NP. Such an argument system allows an efficient prover to prove multiple NP statements, with size smaller than the combined witness length.

We provide the first construction of such an argument system for NP in the common reference string model based on standard cryptographic assumptions. Prior works either require non-standard assumptions (or the random oracle model) or can only support private verification.

At the heart of our result is a new *dual mode* interactive batch argument system for NP. We show how to apply the correlation-intractability framework for Fiat-Shamir – that has primarily been applied to proof systems – to such interactive arguments.

1 Introduction

Consider the following scenario: Alice wants to convince Bob of the veracity of k statements (x_1, \dots, x_k) in an NP language. A naïve solution is for Alice to send a witness w_i for each of the k instances and for Bob to verify each pair (x_i, w_i) . This proof is *non-interactive* (i.e., consists of a single message) as well as *publicly verifiable* (i.e., anyone can verify its correctness). However, it is quite expensive, requiring communication that grows linearly with the combined length of the witnesses.

Can we do better? That is, can we non-interactively prove k NP statements with communication much smaller than $k \cdot m$, where $m = m(|x|)$ is the witness length? Addressing this question is the main focus of this work.

Batch Arguments. We study the problem of designing *batch arguments* (BARG) for NP in the common reference string (CRS) model. Such an argument system allows an efficient prover to compute a non-interactive and publicly verifiable “batch proof” of k NP instances, with size much smaller than $k \cdot m$. If any of the k instances is false, then no polynomial-time cheating prover must be able to produce an accepting proof.

In the *interactive* setting, the problem of batch proofs was first studied by Reingold, Rothblum and Rothblum [48] and more recently in [49, 50]. The focus of these works is on achieving statistical soundness, and we refer the reader to Sect. 1.2 for a discussion. In this work, we focus on the (harder) non-interactive setting but settle for the weaker notion of computational soundness.

Since verifying the membership of k NP instances is itself an NP problem, BARGs with poly-logarithmic communication can be obtained from succinct non-interactive arguments (SNARGs) for NP [3, 4, 18, 28, 44]. However, SNARGs for NP are presently only known to exist under strong, non-falsifiable assumptions [24, 45] (or the random oracle model). In the *designated-verifier* setting, Brakerski, Holmgren and Kalai [6] constructed two-message batch arguments for NP with communication proportional to the size of a single witness, assuming the existence of a computational private information retrieval scheme [13, 41]. The main drawback of their solution is that it requires *private* verification. Recently, Kalai, Paneth and Yang [34] constructed the first non-interactive *publicly verifiable* batch arguments for NP, but rely on a new non-standard (but falsifiable) assumption on groups with bilinear maps.

This state of affairs motivates the following basic question:

Do there exist BARGs for NP based on standard assumptions?

1.1 Our Results

We provide the first construction of a publicly verifiable non-interactive batch argument system for NP in the CRS model from standard computational assumptions. Our scheme achieves non-adaptive computational soundness.

Theorem 1 (Informal). *Let C-SAT be the circuit satisfiability language defined by a boolean circuit $C : \{0, 1\}^{|x|} \times \{0, 1\}^{|y|} \mapsto \{0, 1\}$. Assuming standard computational assumptions, there exists a BARG for C-SAT in the CRS model with non-adaptive soundness. The proof size for k statements is $\tilde{O}((|C| + \sqrt{k|C|}) \cdot \lambda)$, where λ is the security parameter.*

When the number of statements k is smaller than $|C|$, the size of the proof only grows with $|C|$; otherwise, it essentially only grows with k .

On our assumptions. Our construction relies on two essential cryptographic components:

- **Somewhere-Extractable Linearly Homomorphic Commitment.** The first building block for achieving our result is a new notion of *somewhere-extractable linearly homomorphic commitment* (SE-LHC) schemes (Sect. 4). We show an instantiation of SE-LHC assuming the hardness of the quadratic residuosity (QR) assumption.
- **Correlation-Intractable Hash Functions for TC^0 .** Our second cryptographic building block is a correlation-intractable hash function (CIH) [12] for TC^0 circuits. CIH for bounded-depth polynomial-size circuits are known from the learning with errors (LWE) assumption [10, 47]. Very recently, CIH for TC^0 circuits were constructed based on the sub-exponential hardness of the Decisional Diffie-Hellman (DDH) assumption against polynomial-time adversaries [31].

Putting together the above, Theorem 1 can be instantiated based on QR and either LWE or sub-exponential DDH.

We refer the reader to Sect. 2 for an overview of our construction.

On adaptive soundness. Our construction in Theorem 1 achieves non-adaptive (computational) soundness. This seems inherent, as there are known barriers to constructing BARGs with adaptive soundness based on falsifiable assumptions. Specifically, Brakerski, Holmgren and Kalai [6] showed a transformation from *adaptively-sound* BARGs (with argument of knowledge property¹) to adaptively-sound SNARGs using RAM delegation schemes. This in turn allows for using the Gentry-Wichs [24] black-box lower bound for SNARGs.

1.2 Related Works

Batch verification is an interesting question for various cryptographic primitives, and can lead to practical benefits in some settings (see, e.g., [9]).

In the setting of interactive *proofs*, the problem of batch verification of NP has been recently studied in a sequence of works [48–50]. These works consider the class UP, a subset of NP, where each statement in the language has a unique witness of membership. To the best of our knowledge, no positive results are known in this regime for NP. It should be noted that while there are lower bounds on the communication complexity of interactive proofs for languages in NP [25, 26], the lower bounds do not appear to directly extend to the NP batch language $L^{\otimes k}$ due to the additional structure inherent to $L^{\otimes k}$. We refer the reader to [48] for a detailed discussion on this topic.

If we consider computational soundness, where security holds only for computationally bounded cheating provers, Killian’s protocol [39] gives us an interactive batch argument based on collision resistance of hash functions. In the non-interactive setting, Brakerski, Holmgren and Kalai [6] construct *privately-verifiable* non-adaptive batch arguments (of knowledge) assuming computational private information retrieval schemes. Kalai, Paneth and Yang [34] construct a *publicly-verifiable* non-adaptive batch argument, but rely on a new (falsifiable) decisional assumption on groups with bilinear pairings. One can also generically use SNARGs to construct non-interactive batch arguments, but constructions of SNARGs are only known based on strong non-falsifiable assumptions (or in the random oracle mode).

Very recently, there have been works that consider the problem of batch verification for statistical zero-knowledge (SZK) proofs [37, 38]. The specific goals in these works are orthogonal to the problem we consider: the prover in these works is no longer required to be efficient, but it is imperative that the resultant batch protocol is also an SZK proof system.

¹ Our construction in Theorem 1 achieves (non-adaptive) argument of knowledge property.

2 Technical Overview

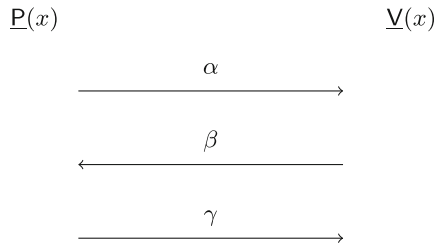
As established in the introduction, we want to design publicly verifiable non-interactive batch arguments for NP. To this end, there exists a well-studied general paradigm one could follow: (i) First, construct an interactive public-coin proof system (P, V) for NP; (ii) Next, apply the Fiat-Shamir (FS) round-collapsing transform [22] on (P, V) with respect to some hash function family \mathcal{H} to obtain a non-interactive proof.

Originally, the soundness of the FS transformation was only established when modeling the hash family as a random oracle. But the transformation has garnered a lot of recent attention with an exciting line of work that demonstrate the soundness of the transformation when the hash function family is *correlation intractable* [12]. In particular, this idea has been used with much success in the context of non-interactive zero-knowledge arguments [7, 10, 11, 16, 17, 29, 31, 35, 47], (publicly verifiable) succinct non-interactive arguments of knowledge for log-space uniform computation [10, 32, 33, 36] and establishing the hardness of the complexity class PPAD [14, 32, 33, 36, 42].

Since this paradigm is central to our work as well, we start by describing the transformation, correlation intractability (CI), and the role it plays in the soundness of the transformation.

2.1 Background

Fiat-Shamir Transformation and CI. The Fiat-Shamir transform with respect to some hash family \mathcal{H} , utilizes a sampled hash function $h \leftarrow \mathcal{H}$ as the common reference string (CRS) to convert a public-coin interactive protocol into a non-interactive proof in the CRS model, where the verifier’s messages are derived (non-interactively) by the prover applying the hash function h to the transcript. For instance, consider the following flow of messages between the prover and the verifier, where the verifier’s message β is a uniformly random string:



The prover computes the verifier’s message as $\beta := h(x, \alpha)$, and the resultant non-interactive proof is the tuple (α, β, γ) - the verifier can recompute β and check if the prover did indeed compute it correctly. Unlike soundness in the interactive setting, where a cheating prover P^* has no control over the verifier’s

message β , in the transformed non-interactive protocol, P^* has *some* control over β . Specifically, P^* can try different values of α to input into the hash function until it gets a β that it considers favorable. Let's formalize what we mean. For a statement $x \notin L$, when the prover evaluates the hash function h on (x, α) , it wants to find an element from the following set of *bad challenges*,

$$\mathcal{B}_{x,\alpha} := \{ \beta \mid \exists \gamma \text{ s.t. } \mathcal{V}(x, \alpha, \beta, \gamma) = 1 \}. \tag{1}$$

Can we hope to enforce some restrictions on the hash family \mathcal{H} , such that it is intractable to find an α such that $h(x, \alpha) \in \mathcal{B}_{x,\alpha}$, i.e. the hash evaluation doesn't result in a *bad challenge*? This is exactly where the correlation intractability of the hash family \mathcal{H} helps. Intuitively, \mathcal{H} is a correlation intractable hash family (CIH) for a function f , if the following holds for all probabilistic polynomial time adversary (PPT) \mathcal{A} ,

$$\Pr_{h \leftarrow \mathcal{H}} [h(x) = f(x) \mid \mathcal{A}(h) = x] \leq \text{negl}(\lambda).$$

This yields the following idea - define a function $\text{BAD}(\cdot)$, that on input (x, α) , outputs an element in $\mathcal{B}_{x,\alpha}$. Let us for the moment assume that $\mathcal{B}_{x,\alpha}$ for all α and $x \notin L$ has at most a single element. If \mathcal{H} is a CIH for $f(\cdot) := \text{BAD}(\cdot)$, then any cheating prover that outputs an accepting transcript (α, β, γ) for $x \notin L$ must break the correlation intractability of \mathcal{H} since $\beta \in \mathcal{B}_{x,\alpha}$ by definition.

But what about when $\mathcal{B}_{x,\alpha}$ consists of multiple elements? We want to argue that the cheating prover doesn't output *any* element from $\mathcal{B}_{x,\alpha}$. If $|\mathcal{B}_{x,\alpha}|$ is *polynomially bounded*, we can argue this via a simple application of the union bound: modify $\text{BAD}(\cdot, \cdot)$ to additionally take in as input an index i , and output the i -th element of $\mathcal{B}_{x,\alpha}$ (for some ordering of the elements). Let $f_i(\cdot) := \text{BAD}(\cdot, i)$, then by the union bound we have for any PPT adversary \mathcal{A} ,²

$$\Pr_{h \leftarrow \mathcal{H}} [h(x) \in \{f_1(x), \dots, f_{|\mathcal{B}|}(x)\} \mid \mathcal{A}(h) = x] \leq |\mathcal{B}| \cdot \text{negl}(\lambda).$$

While our description above is for a protocol with a single verifier message, this can be extended to multi-round protocols by further constraining the interactive protocol to satisfy additional properties such as *round-by-round soundness* [10]. We will elaborate on these properties soon, once we discuss our specific approach.

Clearly, the BAD functions we can support using the above methodology are constrained by the functions for which we can construct CIH. The known CIH from standard assumptions are: bounded-depth polynomial size circuits from LWE [10, 47], linear approximable relations from trapdoor hash functions [7, 21], and TC^0 circuits from sub-exponential DDH [31]. At the very least, we thus require BAD to be efficiently computable.

Putting together the above, we obtain the following design principles for constructing an interactive protocol which is "compatible" with the CIH framework for Fiat-Shamir (w.r.t. known constructions of CIH):

² For notation convenience, we drop the subscript for \mathcal{B} .

1. The BAD function is efficiently computable.
2. For every $x \notin L$ and every α , the size of \mathcal{B} is polynomially bounded.

In this work, we follow the Fiat-Shamir paradigm as well to obtain our main result. In the following, we start by discussing potential choices for an interactive protocol that meets our desired efficiency goals while still being compatible with the CIH framework.

Considerations for the interactive protocol. Since the Fiat-Shamir transformation does not reduce the communication complexity of the interactive protocol, our starting point needs to be a protocol where the total communication between the prover and verifier is much smaller than $O(km)$, where k denotes the number of instances, and m the length of a single witness. A natural candidate that satisfies our requirements is Killian’s protocol for languages in NP [39]. Specifically, it is a public-coin interactive protocol where the total communication between the prover and verifier is significantly smaller than the length of the witness. Thus by defining the following NP language, $L^{\otimes k} = \{(x_1, \dots, x_k) : \forall i \in [k], x_i \in L\}$, Killian’s protocol gives us a public coin interactive argument with total communication significantly smaller than $O(km)$. Unfortunately, a recent work of [2] established non-trivial barriers towards instantiating the hash function in the Fiat-Shamir transformation applied to Killian’s protocol.

There is in fact a broader point to consider: Killian’s protocol is an *argument*, i.e. its soundness holds only against computationally bounded cheating provers. In general, successful applications of the Fiat-Shamir paradigm when used in conjunction with CIH, have been largely restricted to interactive *proofs*, where the soundness holds against computationally unbounded cheating provers. Intuitively, this is because \mathcal{B} , as defined in Eq. 1, does not capture the computational resource bounds of a cheating prover. Specifically, \mathcal{B} may contain exponentially many elements but does not capture the fact that for a computationally bounded cheating prover, finding the γ corresponding to $\beta \in \mathcal{B}$ is intractable. And as we have already outlined above, we need \mathcal{B} to be of polynomial size. In fact, there are examples of certain interactive arguments that are not sound on the application of the Fiat-Shamir transformation (see e.g. [1, 27]).

Given the above state of affairs, the natural approach is to consider public coin interactive batch *proofs* for NP that achieve the same succinctness properties as (non-interactive) BARGs. Presently, however, interactive batch proofs are only known for the class UP, a subset of NP for which there is exactly one witness of membership for each statement [48–50]. Indeed, constructing such proofs for NP is an open problem.

2.2 Dual-Mode Interactive Batch Arguments

We therefore deviate from the above approach and instead define and construct a primitive we call *dual-mode interactive batch arguments*. Intuitively, these are interactive *arguments* in the common reference string (CRS) model, where the CRS can be generated in two computationally indistinguishable modes - (1)

normal mode; and (2) *trapdoor mode*. We require that in the trapdoor mode, the protocol is sound against all (possibly unbounded) cheating provers; however, in the normal mode, it only achieves computational soundness.

This gives us the best of both worlds – we bypass the problem of constructing interactive batch proofs, but still retain the possibility of applying the Fiat-Shamir transform to the protocol when it is executed in the trapdoor mode (without running into the issues that arise for arguments). In order to apply the Fiat-Shamir transform, we require some additional properties from dual mode interactive batch arguments: specifically, we require such protocols to be *Fiat-Shamir friendly*, a notion we will elaborate on shortly.

We present a dual-mode interactive batch argument system for proving multiple instances of the NP-complete problem R1CS. An R1CS instance \mathfrak{x} is defined to be the tuple $\mathfrak{x} := (A, B, C, \text{io}, m)$, where io denotes the public input and output of the instance, and $A, B, C \in \{0, 1\}^{m \times m}$ are matrices. We say that a vector $w \in \{0, 1\}^{m - |\text{io}| - 1}$ is a witness for \mathfrak{x} if $(A \cdot z) \circ (B \cdot z) = (C \cdot z)$, where $z = (\text{io}, 1, w)$, \cdot is the matrix-vector product, and \circ is the Hadamard (entry-wise) product.³

Background: Spartan Protocol. Our starting point is the Spartan protocol [51] which proves the satisfiability of a *single* R1CS instance \mathfrak{x} with total communication sub-linear in the witness size $|w|$, i.e. the protocol is succinct. The Spartan protocol is defined over a field \mathbb{F} , such that $\log |\mathbb{F}| \approx \lambda$, and follows roughly the structure described below:

1. The prover first computes a commitment c to the witness w , that it sends to the verifier. In order to achieve communication succinctness, $|c|$ must be sub-linear in m . (We shall see below that the commitment scheme needs to satisfy some additional properties.)
2. The verifier then sends a random element $\tau \in \mathbb{F}^s$, where s is such that $m = 2^s$.
3. It was shown in [51] that with probability $s/|\mathbb{F}|$ over the choice of τ , any R1CS instances can then be reduced to the following check:

$$\sum_{x \in \{0,1\}^s} \mathcal{G}_{\text{io},w,\tau}(x) = 0, \tag{2}$$

where $\mathcal{G}_{\text{io},w,\tau} : \mathbb{F}^s \mapsto \mathbb{F}$ is a polynomial with degree 3 in each variable, and is determined entirely by \mathfrak{x} , the witness w and τ . For the purpose of our discussion, the exact form of the polynomial is not immediately relevant. Note that without the witness w , the verifier does not have a representation of $\mathcal{G}_{\text{io},w,\tau}$, but we shall see shortly that it doesn't matter.

The above check is precisely the scenario where the *sumcheck protocol* [43, 52], an interactive protocol between a prover and verifier, is useful. In the sumcheck protocol, the prover is attempting to convince the verifier of the

³ R1CS instances are more generally defined over a field, but for this overview we will consider them over \mathbb{F}_2 (or $\{0, 1\}$). An instance of Boolean circuit satisfiability (C-SAT), defined by a circuit C can be transformed to an R1CS instance where $m \approx |C|$. See the full version for details on the transformation.

claim $\sum_{b_1, \dots, b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell) = v$, where $g : \mathbb{F}^s \mapsto \mathbb{F}$ is an s variate polynomial of degree at most d in each variable, and $v \in \mathbb{F}$ is a publicly known value. The resultant interactive protocol is an s round public coin *proof* where the prover sends $O(d \cdot s)$ field elements. Importantly the verifier is only required to evaluate g at a *single point* $r^* \in \mathbb{F}^s$ at the end of the protocol, where r^* determined solely by the verifier’s randomness in sumcheck protocol.

4. The prover and verifier run the sumcheck protocol for Eq. 2, at the end of which verifier needs to evaluate $\mathcal{G}_{\text{io},w,\tau}(\cdot)$ at the point r^* (and compare against some value determined by the sumcheck protocol). But since the verifier does not have access to $\mathcal{G}_{\text{io},w,\tau}(\cdot)$, it asks the prover to send relevant information so that it can complete the check. Since the Spartan protocol requires this message from the prover to be succinct, the prover cannot send w in the clear.

Fortunately, it turns out that the value that the prover needs to send is simply a linear combination of the bits of w where the linear coefficients are determined entirely by A, B, C, τ and r^* i.e. let $\sum_{i \in [m]} \sigma_i \cdot w_i$ be the corresponding linear combination where the coefficients σ_i are known to both the prover and verifier, and are even independent of io .⁴

5. The prover now opens the commitment c to $\sum_{i \in [m]} \sigma_i \cdot w_i$ such that the opening is succinct. This allows the verifier to complete its check.

Spartan provides various instantiations for the commitment scheme satisfying the above properties, where the commitment opening is an interactive protocol. The resulting protocol is computationally sound.

The Spartan protocol does not satisfy our desired properties from a dual-mode interactive batch argument. However, it serves as a useful starting point for us. In Spartan, the goal was to have the total communication be sub-linear in m , while in the batch setting, we are fine with total communication proportional to a *single witness*. This in turn means that we can consider commitment schemes where the commitment size is proportional to a single witness. Let us now see how we can use this insight to adapt the Spartan protocol to both make it suitable for batch verification, and achieve the notion of dual-mode batch arguments.

Our Construction. We now discuss the main steps in our interactive protocol, while highlighting the differences from the above discussion. We want to batch prove k instances $\{\mathbf{x}^{(j)}\}_{j \in [k]}$ where the matrices A, B and C are the same across all instances, and only the public input-output io varies across the instances. The reader may view this as multiple instances with the same relation circuit, but different statements. The description of the protocol now follows.

1. To commit to a batch of witnesses $\{w^{(j)}\}_{j \in [k]}$, we follow the batch commitment strategy in [48]: arrange the witnesses as rows of a $k \times m$ matrix, and commit to the column of each matrix, i.e. $\forall i \in [m], c_i \leftarrow \text{Com}(w_i^{(1)}, \dots, w_i^{(k)})$. If the k -tuple commitment has size $O(\lambda)$, then the total commitment is of size $\tilde{O}(m)$, ignoring polynomial factors in $O(\lambda)$.

⁴ Strictly speaking, the prover needs to send 3 separate linear combinations of the witness, but we ignore this here for simplicity.

This indicates that our *commitment scheme must allow us to commit to the k -tuple succinctly.*

2. Given that each instance has a different statement io and witness w , each of the k instances define a different polynomial, giving rise to the k polynomials $\{\mathcal{G}_{io,w,\tau}^{(j)}\}_{j \in [k]}$. The prover and verifier then run k sumcheck protocols in parallel with the *same verifier randomness*. As discussed earlier, at the end of the sumcheck protocols, the verifier needs to evaluate each of these polynomials at points $r^{*(j)}$ determined solely by the verifier’s randomness in the sumcheck protocol.

Since the verifier uses the same randomness across all instances of the sumcheck protocol execution, the polynomials need to be evaluated at the *same point* r^* . Additionally, since the linear coefficients depend only on A, B, C, τ and r^* , this in turn implies that the linear coefficients for all the witnesses $w^{(j)}$ are the same: $(\sigma_1, \dots, \sigma_m)$.

3. As in Spartan, the prover now needs to send $\sum_{i \in [m]} \sigma_i^{(j)} w_i^{(j)}$ to the verifier. For convenience, this can be re-written as sending the k -tuple, $\sum_{i \in [m]} \sigma_i \cdot (w_i^{(1)}, \dots, w_i^{(k)})$, where \cdot indicates component-wise multiplication.

If our commitment scheme satisfies linear homomorphism, i.e.

$$\text{Com}\left(\sum_{i \in [m]} \sigma_i \cdot (w_i^{(1)}, \dots, w_i^{(k)})\right) = \sum_{i \in [m]} \sigma_i \text{Com}(w_i^{(1)}, \dots, w_i^{(k)}),$$

then it suffices for the prover to open to the commitment $\sum_{i \in [m]} \sigma_i c_i$.

Thus our *commitment scheme must satisfy linear homomorphism (as described above), with the size of the opening proportional to the size of the underlying message.*

Let us go back to our requirement from dual-mode interactive batch arguments. For the protocol to achieve statistical soundness in the trapdoor mode, we need at the very least, the commitment to be statistically binding. However, this seems at odds with our succinctness requirements since we want the total number of bits sent to be significantly smaller than the size of the message committed.

Key Tool: Somewhere-Extractable Linearly Homomorphic Commitments.

We resolve this issue by utilizing a commitment scheme in the CRS model, where the CRS is generated in one of two *computationally indistinguishable ways* - (1) *normal mode*; or (2) *extraction mode*. In the *extraction mode*, the CRS generation algorithm takes as input an index i^* , and additionally outputs an *extraction trapdoor* td that is not a part of the CRS. We require that the commitment of the k -tuple in the *extraction mode* for index i^* , is statistically binding at the i^* -th index of the commitment. Further, there is an efficient algorithm Ext such that given the trapdoor td , Ext extracts the underlying message at the i^* -th index, and this holds even if the commitment was “malformed”. Additionally, the extraction also satisfies *linear homomorphism*, i.e. $\sigma_1 \cdot \text{Ext}(c_1, td) + \sigma_2 \cdot \text{Ext}(c_2, td) = \text{Ext}(\sigma_1 \cdot c_1 + \sigma_2 \cdot c_2, td)$. The linear homomorphism

property of extraction ensures that once we extract from the commitments, the opening of the linear homomorphic evaluation can be computed solely from the linear coefficients - ensuring that the committer is bound to opening of the linear homomorphism.

If all m commitments are committed via the *extraction mode* CRS for index i^* , then the prover is statistically bound to $w^{(i^*)}$. Then intuitively, in the extraction mode, the security can be reduced to the soundness of the other components of the protocol for the i^* -th instance. The reduction to the check for polynomial $\mathcal{G}_{\text{io},w,\tau}^{(i^*)}$ (via [51]), and the sumcheck protocol are both statistically sound, thereby satisfying overall statistical soundness. Thus, by setting the *trapdoor mode* (resp., normal model) CRS to be the extraction mode (resp., normal mode) CRS of the commitment scheme, we obtain a dual-mode interactive batch argument. Note the added syntax for the *trapdoor mode* of the dual-mode interactive batch argument - it takes in as input an index i^* , and generates a trapdoor td (not be included in the CRS).

We now summarize our requirements of the commitment scheme from the above discussion:

1. Commitment scheme for k -tuples in the CRS model, with indistinguishable methods of generating the CRS - *normal mode* or *extraction mode* such that the commitment is statistically binding at the i^* -th index when the CRS is generated in the extraction mode on input i^* .
2. Efficient extraction of the message at i^* -th index in the extraction mode, given the trapdoor td .
3. The commitment should allow for linear homomorphism (even over the extracted values).
4. The commitment should be succinct, while the opening should depend only on the size of the committed message.

We refer to such commitments as *somewhere-extractable linearly homomorphic commitments*. Our notion is similar to the notion of somewhere statistically-binding hash functions [30], but requires some additional properties. Later, in Sect. 2.4, we describe our construction of such commitment schemes based on the quadratic residuosity assumption. For now, we will simply assume that such commitment schemes exist.

One point of note is that since the CRS of the commitment scheme requires the index of the statement we want to prove soundness for, we can only achieve *non-adaptive security*. We will later show in the technical sections that this is in some sense the best that one can hope for.

Costs. From the description of the protocol, the communication cost for the commitment (and its opening) is $\tilde{O}(m)$, while the communication cost from k sumcheck protocols is $\tilde{O}(ks) = \tilde{O}(k \log m)$, giving us a total communication cost of $\tilde{O}(m + k \log m)$.

2.3 Fiat-Shamir Compatibility

As we have alluded to before, constructing a dual-mode interactive batch argument is an important first step towards a non-interactive protocol. But by itself, it is not enough. We need to show that our constructed protocol is Fiat-Shamir friendly. This has been recently formalized by [32] as the notion of *Fiat-Shamir (FS) compatibility*, that extends our earlier discussion in Sect. 2.1 on the relationship between CIH and the Fiat-Shamir transform.

Let the prover’s i -th message in the protocol be denoted by α_i , while the corresponding verifier message by β_i . The protocol transcript trans_i is defined to be $\text{trans}_i := (\alpha_1, \beta_1, \dots, \alpha_i, \beta_i)$, which collects all messages up to (and including) the i -th round messages. An interactive proof is said to be FS compatible if it follows the following two properties:

Round-by-round soundness: There is a function State that takes as input the statement x , and a transcript prefix $\text{trans}_i := (\alpha_1, \beta_1, \dots, \alpha_i, \beta_i)$, and outputs Accept or Reject . We require some additional properties from State : for every $x \notin L$, $\text{State}(x, \emptyset) = \text{Reject}$, and for every full transcript trans the verifier rejects if $\text{State}(x, \text{trans}) = \text{Reject}$. Perhaps, most importantly, we require that if $\text{State}(x, \text{trans}) = \text{Reject}$, then for *any* prover message α , $\text{State}(x, \text{trans}|\alpha|\beta) = \text{Reject}$ with overwhelming probability over the choice of β .

Efficient BAD function: For every $x \notin L$, when $\text{State}(x, \text{trans}_i) = \text{Reject}$, we require an efficiently computable function BAD ⁵ that outputs the “bad” verifier challenges β that will result in State switching output to Accept , i.e. if $\text{State}(x, \text{trans}) = \text{Reject}$, then $\text{BAD}(x, \text{trans}|\alpha)$ outputs a uniformly random element from the set \mathcal{B} defined as

$$\mathcal{B} := \{ \beta \mid \text{State}(x, \text{trans}|\alpha|\beta) = \text{Accept} \}.$$

From our earlier design principles, we require the size of the set \mathcal{B} to be polynomially bounded.

Before we proceed, let’s recall our discussion from Sect. 2.1 on Fiat-Shamir and CI-hash functions. It is easy to see that the discussion there also applies here - as long as we can construct a CIH \mathcal{H} that is CI for the circuits computing BAD , then the Fiat-Shamir transformed protocol with respect to \mathcal{H} is sound.

We know of the following CI-hash functions based on standard assumptions⁶:

- CI for all a priori polynomially bounded circuits assuming LWE [47]; and
- CI for all of TC^0 assuming sub-exponential security of DDH [31].

⁵ Unlike the definition in [32], we will require any non-uniform advice to the BAD function to also be *efficiently* computable.

⁶ We note that the CI-hash function constructed in [7] is also based on standard assumptions, but the class of functions that it supports (i.e. class it is CI for) is very small, and therefore limits its applicability.

We want to be able to leverage both of these constructions for our final non-interactive batch argument. Since the size (and depth) of the circuit computing BAD directly corresponds to the functions for which we need CI, to achieve a result based on sub-exponential security of DDH, we need to show that the function BAD can be computed in TC^0 . We call such protocols to be *strongly FS compatible*.

Let us now demonstrate that our dual-mode protocol in the trapdoor mode is FS-compatible. Recall that in the trapdoor mode an index i^* is specified, and we shall prove FS compatibility when $\mathfrak{x}^{(i^*)} \notin L_{\text{R1CS}}$. This is sufficient, since for a batch instance to be false, there is at least one index j such that $\mathfrak{x}^{(j)} \notin L$. In particular, this allows us to ignore the other sumcheck executions while establishing FS compatibility. We will further show that BAD can also be computed in TC^0 . Since we only focus on a single instance $\mathfrak{x}^{(i^*)}$, in what follows, we skip the index i^* for the instance to simplify notation.

Round-by-Round soundness. The verifier messages can be split into two cases: (a) $\tau \in \mathbb{F}^s$; (b) verifier messages inside the sumcheck protocol. We only sketch here the main ideas and refer the reader to the technical sections for more details as our primary focus will be on the construction of the BAD function.

For the sumcheck, we rely on [32] that already establishes the sumcheck protocol to be round-by-round sound. The main difference is that [32] requires full knowledge of the polynomial over which the sumcheck is computed. In our setting, however, the polynomial $\mathcal{G}_{i_0, w, \tau}$ is (partially) determined by the witness, which is sent within the commitment. We resolve this issue by using the trapdoor td to extract the i^* -th witness and compute the polynomial, since the CRS was generated in the trapdoor mode for i^* . For the verifier message τ , we can rely on the Theorem underlying Spartan [51] that shows that any R1CS instance \mathfrak{x} can be reduced to the sum $\sum_{x \in \{0,1\}^s} \mathcal{G}_{i_0, w, \tau}(x) = 0$ other than with probability $s/|\mathbb{F}|$ over the choice of τ . The actual State computation for τ will be elaborated upon in the BAD function computation below.

Efficient BAD function. As described above, verifier messages can be split into two cases. From the definition of the BAD function, it suffices to build two separate functions, one for each cases. Let's start with the simpler case of the sumcheck verifier messages.

Sumcheck BAD function: In the sumcheck protocol, for each round $i \in [s]$, the prover sends a univariate polynomial $g_i^* : \mathbb{F} \mapsto \mathbb{F}$ of degree 3 to the verifier. If computed correctly, it should correspond to the polynomial g_i , defined as

$$g_i(x) := \sum_{x_{i+1}, \dots, x_s \in \{0,1\}} \mathcal{G}_{i_0, w, \tau}(\beta_1, \dots, \beta_{i-1}, x, x_{i+1}, \dots, x_s).$$

The set of bad challenges in the i -th round are the verifier challenges β_i such that both polynomials g_i and g_i^* evaluate to the same value on β_i , i.e. $\mathcal{B} := \{\beta_i \mid g_i(\beta_i) = g_i^*(\beta_i)\}$. Alternatively \mathcal{B} consists of the roots of the polynomial $g_i - g_i^*$. Since $\mathcal{G}_{i_0, w, \tau}$ is a polynomial that is degree 3 in each variable, $|\mathcal{B}| \leq 3$.

Unlike [32], which demonstrate BAD function for the general sumcheck, we focus on the setting where the *true* polynomial g_i can be computed in polynomial time (e.g. $s = O(\log \lambda)$). Thus on input, $(\mathbf{x}, \text{trans}_{i-1}|\alpha_i)$, BAD (i) parses α_i as the polynomial g_i^* ; (ii) computes the *true* polynomial g_i , using the trapdoor first to extract w and determine $\mathcal{G}_{\text{io},w,\tau}$; and (iii) use a polynomial time algorithm like Cantor-Zassenhaus to compute the (three) roots of $g_i - g_i^*$, and output one at random.

τ BAD function: To describe the BAD function corresponding to τ , we need to look at the polynomial $\mathcal{G}_{\text{io},w,\tau}$ implied by [51] (Theorem 2). So far we have focused on $\mathcal{G}_{\text{io},w,\tau}(x)$ as a polynomial over the variables x , with $\tau \in \mathbb{F}^s$ fixed. Let us now focus on the same polynomial over both x and τ , i.e. for every τ , $\mathcal{G}_{\text{io},w,\tau}(x) = \mathcal{G}'_{\text{io},w}(x, \tau)$. In fact [51] showed that $\mathcal{G}'_{\text{io},w}(x, \tau)$ is a polynomial over x_1, \dots, x_s and τ_1, \dots, τ_s that has degree 1 in each τ_i (see full version for details). Thus, we can rewrite $\sum_{x \in \{0,1\}^s} \mathcal{G}_{\text{io},w,\tau}(x)$ as a polynomial over τ_1, \dots, τ_s . Specifically, let

$$Q(\tau) := \sum_{x \in \{0,1\}^s} \mathcal{G}'_{\text{io},w}(x, \tau),$$

where Q is a polynomial over s variables τ_1, \dots, τ_s , with degree 1 in each τ_i . Note that as in the case of $\mathcal{G}_{\text{io},w,\tau}$, Q is determined by the witness w that only the prover has access to. For \mathbf{x} and w such that $\mathcal{R}_{\text{R1CS}}(\mathbf{x}, w) = 1$, the correctly computed polynomial $Q_{\text{io},w}$ is the zero polynomial, i.e. $Q_{\text{io},w} \equiv 0$. The random $\tau \in \mathbb{F}^s$, sent by the verifier is to test whether $Q(\tau) = 0$. If $Q \not\equiv 0$, then by the Schwartz-Zippel lemma, $Q(\tau) = 0$ with probability at most $s/|\mathbb{F}|$ over the choice of τ , which is negligible in λ for our choice of \mathbb{F} . This suggests the following strategy for BAD, when $Q \not\equiv 0$, let $\mathcal{B} := \{\tau \in \mathbb{F}^s \mid Q(\tau) = 0\}$. BAD then works as follows: (i) uses the trapdoor td to first extract w and determine Q ; and (ii) solve for τ from \mathcal{B} and output a random such τ .

While this appears to work on the surface, on closer inspection it can be observed that while the Schwartz-Zippel lemma guarantees the probability to be at most $s/|\mathbb{F}|$, the size of the set \mathcal{B} can be exponential ($|\mathcal{B}| \approx |\mathbb{F}^{s-1}|$). As indicated by our design goals at the start, this is undesirable and something we do not know how to work around.

We take an alternate approach. Instead of using a single hash function that outputs the vector $\tau \in \mathbb{F}^s$, we consider a sequence of hash functions (h_1, \dots, h_s) that each output a single τ_i . Specifically, for every i , $\tau_i := h_i(\mathbf{x}, \tau_1, \dots, \tau_{i-1})$.

Let $Q_{|\tau_1^*, \dots, \tau_{i-1}^*}$ be the polynomial Q with the first $i - 1$ variables fixed to be values $\tau_1^*, \dots, \tau_{i-1}^*$. If $Q \not\equiv 0$, then we want it to continue to be the case that for the prefix $\tau_1^*, \dots, \tau_{i-1}^*$, $Q_{|\tau_1^*, \dots, \tau_{i-1}^*} \not\equiv 0$. This then lets us define the i -th bad set \mathcal{B}_i when $Q_{|\tau_1, \dots, \tau_{i-1}} \not\equiv 0$,

$$\mathcal{B}_i := \left\{ \tau \in \mathbb{F} \mid Q_{|\tau_1, \dots, \tau_{i-1}, \tau} \equiv 0 \right\}.$$

Before we describe the BAD function, let us take a moment to see how one determines whether $Q_{|\tau_1, \dots, \tau_{i-1}, \tau} \equiv 0$. This corresponds to all coefficients of the

said polynomial to be 0. At a high level, from the description of Q , the coefficients are determined by the sum over $m = 2^s$ values, which in turn is computable in polynomial time as $m = \text{poly}(\lambda)$. Then a bad τ simply corresponds to those elements in \mathbb{F} that result in the coefficients becoming 0. Since the polynomial is *linear* in each variable, solving for such a τ corresponds to solving a linear system in \mathbb{F} . Correspondingly, for all i , the set \mathcal{B}_i is of bounded polynomial size. We refer the reader to the full version for more details on these steps.

We are finally in a position to describe the BAD function, which on input $(\mathbb{x}, \tau_1, \dots, \tau_{i-1})$ (note that the prover message is empty) does the following: (i) use the trapdoor td to first extract w and determine Q , and then correspondingly $Q|_{\tau_1, \dots, \tau_{i-1}, \tau}$; and (ii) solve the linear equation in τ such that $Q|_{\tau_1, \dots, \tau_{i-1}, \tau} \equiv 0$, and output such a τ if it exists.

From our discussions above, BAD is in fact efficiently computable, and thus satisfies our requirement.

BAD has low depth. To base our non-interactive protocol on CIH for TC^0 , we need to demonstrate that the BAD function for both cases can be computed in TC^0 . In contrast to when we established that BAD was efficient, here, the simpler case is the BAD function for τ . But before we proceed, we note that in both cases, we require trapdoor extraction, and thus we additionally require *low-depth extraction* property from our commitment scheme. We proceed with our discussion assuming this to be the case, and will provide more details when discussing out construction of the commitment scheme in Sect. 2.4.

τ BAD function: In the above description, we are only solving linear equations in \mathbb{F} , which can be computed in TC^0 , thus trivially giving us the required property.

Sumcheck BAD function: Unfortunately, things are not so simple for the BAD function in the sumcheck case. The BAD function as described, needs to compute a root of a degree 3 polynomial in \mathbb{F} . While we do know how to do this in polynomial time, for computing roots in low depth, we are only aware of root finding for degree 2 polynomials in \mathbb{F} to be in TC^0 .

To circumvent this issue, we take a closer look at the polynomial $\mathcal{G}_{\text{io}, w, \tau}$ ⁷.

It turns out that $\mathcal{G}_{\text{io}, w, \tau}$ is of a special form (see full version for details), where we compute a sumcheck protocol for,

$$\sum_{x \in \{0,1\}^s} \mathcal{G}_{\text{io}, w, \tau}(x) = \sum_{x \in \{0,1\}^s} f_{\text{io}, w, \tau}(x) \left(\prod_{j=1}^s h_{j, \tau}(x_j) \right) = 0,$$

where f is a polynomial with individual degree 2, and each $h_{j, \tau}$ is a univariate polynomial in x_j with degree 1. Moreover, the coefficients of $h_{i, \tau}$ are determined only by τ , and therefore known to the verifier once it samples τ . This suggests

⁷ For simplicity, we focus on a single polynomial here as our explanation extends to the batch setting too.

a slight modification of the sumcheck polynomial, where the prover in the i -th round sends the degree 2 polynomial $g^{*'}_i$ to the verifier which it has purportedly computed as,

$$g'_i(x) = \sum_{x_{i+1}, \dots, x_s \in \{0,1\}} f_{i\circ, w, \tau}(\beta_1, \dots, \beta_{i-1}, x, x_{i+1}, \dots, x_s) \left(\prod_{j=1}^{i-1} h_{j, \tau}(\beta_j) \right) \left(\prod_{j=i+1}^s h_{j, \tau}(x_j) \right).$$

The verifier then locally computes $h_{i, \tau}$, and computes g_i . Clearly, the three roots of the polynomial $g_i^* - g_i$ consist of the two roots of $g_i^{*'} - g_i$ and the root of $h_{i, \tau}$. Thus, by modifying the sumcheck protocol as suggested above, we can then reduce the root computation in BAD to computation of roots for a degree 2 polynomial, and a degree 1 polynomial, both of which we can compute in TC^0 .

This establishes that our dual-mode protocol in the trapdoor mode is strongly FS-compatible. [32] demonstrate that the Fiat-Shamir transformation with respect to \mathcal{H} for any FS-compatible protocol is sound as long as \mathcal{H} is CI for polynomial size functions (larger than BAD). We extend their proof to demonstrate that if we strengthen FS compatibility to strong FS compatibility, it suffices for \mathcal{H} to be CI for TC^0 .

Next, we show how to leverage our protocol to construct a non-interactive batch argument (BARG).

Going from FS-Compatibility to BARGs. In this final step, we finally construct our non-interactive arguments. We apply the Fiat-Shamir transform to the dual-mode interactive batch argument to achieve a publicly verifiable non-adaptive BARG in the CRS model. For soundness of the transform we rely on (i) *mode indistinguishability* property of the protocol to switch to the trapdoor mode; and (ii) then in the *trapdoor mode*, we rely on the FS-compatibility that we have discussed above.

Communication sub-linear in k . The above construction has an additive term that is linear in k (recall that the communication cost is $\tilde{O}(m + k \log m)$). We describe how one can generically make this sub-linear by using fairly standard techniques. The idea is to simply batch k_1 instances into a larger instance of the language $L^{\otimes k_1} := \{(x_1, \dots, x_{k_1}) : \forall i \in [k_1], x_i \in L\}$ that has a relation circuit of size $k_1|C| + k_1$, where $|C|$ is the size of the underlying relation circuit. Then we apply our dual-mode batch argument for k/k_1 instances of $L^{\otimes k_1}$. By setting $k_1 \approx O(\sqrt{k})$, we get communication that is sub-linear in k . Note that from our earlier discussion $m \approx k_1|C| + k_1$.

2.4 Somewhere-Extractable Linearly Homomorphic Commitment

We now finally describe our construction of the somewhere-extractable linearly homomorphic commitment scheme. Over the course of the above discussion, we have accumulated various requirements that our commitment scheme must

satisfy. We describe a construction that achieves these properties based on the *quadratic residuosity* (QR) assumption.

We start by focusing on the simpler goal of constructing a somewhere statistically binding commitment scheme building on ideas from the recent work on trapdoor hash functions [21].⁸ We will discuss how to achieve the extraction and linear homomorphism properties later.

Recall that, for any Blum integer $N = p \cdot q$, where p, q are primes such that $p \pmod 4 = q \pmod 4 = 3$, we denote \mathbb{Z}_N^* as the multiplicative group modulo N , and \mathbb{J}_N as the subgroup of \mathbb{Z}_N^* with Jacobi symbol $+1$, and \mathbb{QR}_N be the subgroup of quadratic residues. Let $\mathbb{H} = \{-1, +1\}$ also be a multiplicative group, then $\mathbb{J}_N = \mathbb{H} \times \mathbb{QR}_N$. We now describe the commitment scheme:

- The *trapdoor* mode commitment key for the coordinate i^* consists of two arrays of group elements.

$$\begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} g_1 & g_2 & \dots & g_{i^*} & \dots & g_k \\ g_1^s & g_2^s & \dots & -g_{i^*}^s & \dots & g_k^s \end{bmatrix},$$

where $s \leftarrow \lfloor (N-1)/2 \rfloor$ is sampled uniformly at random, and the elements of the second row are the corresponding first row elements raised to the exponent s , except that we flip the sign on the i^* -th coordinate.

In the *normal* mode, we do not flip the sign, i.e. let $\mathbf{h} = (\mathbf{g})^s$. The mode indistinguishability relies on the quadratic residuosity assumption⁹.

- To commit to a vector $\mathbf{x} = (x_1, x_2, \dots, x_k)$ of length k , we compute $(c_g = \prod_{i=1}^k g_i^{x_i}, c_h = \prod_{i=1}^k h_i^{x_i})$. Then $c_h = c_g^s \cdot (-1)^{x_{i^*}}$. Hence, x_{i^*} is statistical binding. Furthermore, the commitment size is compact, since it only contains two group elements.

Linear Homomorphism and Extraction. We now discuss how to achieve the desired extraction and the linear homomorphism properties. We observe that the commitment described above is essentially an encryption of x_{i^*} . Hence, one can use the trapdoor $\text{td} = (p, q, s)$ to extract x_{i^*} . The linear homomorphism works as follows: if we denote the commitment of \mathbf{x} under the key (\mathbf{g}, \mathbf{h}) as $(\mathbf{g}^{\mathbf{x}}, \mathbf{h}^{\mathbf{x}})$, then for any two commitments $(\mathbf{g}^{\mathbf{x}}, \mathbf{h}^{\mathbf{x}})$, $(\mathbf{g}^{\mathbf{y}}, \mathbf{h}^{\mathbf{y}})$, and any integers $a, b \in \mathbb{Z}$, we can compute

$$((\mathbf{g}^{\mathbf{x}})^a \cdot (\mathbf{g}^{\mathbf{y}})^b = \mathbf{g}^{a \cdot \mathbf{x} + b \cdot \mathbf{y}}, \quad (\mathbf{h}^{\mathbf{x}})^a \cdot (\mathbf{h}^{\mathbf{y}})^b = \mathbf{h}^{a \cdot \mathbf{x} + b \cdot \mathbf{y}}),$$

which is exactly the commitment for $a \cdot \mathbf{x} + b \cdot \mathbf{y}$.

However, if we use the above commitment scheme for our application to batch arguments, we face the following challenge: the field operation needs modulo 2 computation, but the honest prover can not hope to perform such computation, since the homomorphic operation is over \mathbb{Z} .

⁸ Similar ideas have also been used in the constructions of somewhere statistically-binding hash functions [30, 40, 46] and hash encryption schemes [8, 19, 20, 23].

⁹ The mode indistinguishability follows from [5], which relies on the quadratic residuosity assumption.

To overcome this issue, we have the honest prover do all the operations over the polynomial ring $\mathbb{Z}[\alpha]$, instead of the field \mathbb{F} . Note that this modification does not affect completeness since the honest prover is essentially proving some polynomial identities (e.g. the R1CS instance $(A \cdot z) \circ (B \cdot z) = C \cdot z$ reduced from circuit satisfiability), and such identities hold regardless of whether the underlying variables are taken from a field or a ring. For soundness, we make the following observation: if a proof is accepted over the ring $\mathbb{Z}[\alpha]$, then if we further perform modulo 2 operation, the proof must still be accepted. Hence the soundness can be reduced to the case when operations are over \mathbb{F} . See Sect. 5 for a more detailed discussion.

(Linearly Homomorphic) Extraction from *any* Commitment. The aforementioned extraction and linear homomorphism only works for “well-formed” commitments. In order to prove round-by-round soundness of our dual-mode interactive batch argument protocol, however, we need the extraction works for *any* (possibly not well-formed) commitment. Moreover, the linear homomorphism property must also hold over the extracted values.

To achieve such a property, we observe that for any (possibly not well-formed) commitments $c = (c_g, c_h) \in \mathbb{J}_N \times \mathbb{J}_N$, we can still compute c_h/c_g^s , which is also a group element in \mathbb{J}_N . From the decomposition $\mathbb{J}_N = \mathbb{H} \times \mathbb{QR}_N$, there exists a unique $m \in \mathbb{Z}_2$ and $g \in \mathbb{QR}_N$ such that $c_h/c_g^s = (-1)^m \cdot g$. Hence, we define the extracted message for c as m . Since N is a Blum integer, $|\mathbb{QR}_N| = (p-1)/2 \cdot (q-1)/2$ is an odd number. We let n denote $|\mathbb{QR}_N|$. Then, we extract m by computing

$$(c_h/c_g^s)^n = (-1)^m \cdot g^n = (-1)^m.$$

We show that this extraction can be decomposed to an off-line pre-computation phase and an online extraction phase, where the online extraction can be computed in TC^0 . We allow the off-line pre-computation to be deeper than TC^0 circuits, since in our protocol, the pre-computation is always performed honestly by the prover and the verifier.

We now show that the linear homomorphism property also holds for the above extraction algorithm. For any two commitments $c = (c_g, c_h), d = (d_g, d_h)$, the extraction is a “linear operation” over c_g, c_h , i.e. if $\text{Ext}(c, \text{td}) = m_c$, then $(-1)^{m_c} = c_h^n c_g^{-sn}$. Similarly, if $\text{Ext}(d, \text{td}) = m_d$, then $(-1)^{m_d} = d_h^n d_g^{-sn}$. Now for any linear combination $a, b \in \mathbb{Z}$, when we extract from $a \cdot c + b \cdot d$, we compute $(c_h^a \cdot d_h^b)^n \cdot (c_g^a \cdot d_g^b)^{-sn} = (-1)^{a \cdot m_1 + b \cdot m_2}$. Hence, the extracted value for $a \cdot c + b \cdot d$ is $a \cdot m_1 + b \cdot m_2 \pmod{2}$, which establishes the linear homomorphism property.

For more details, see Sect. 4.2.

Full Version. Due to space constraints, preliminaries and details of the proofs have been omitted from this manuscript, and can be found in the full version of the paper [15].

3 Preliminaries

We defer most of the preliminaries to the full version, but describe here some notation that will be used in the rest of the paper.

We start with some basic notation: For any n length string a , we denote by a_i the i -th position of the string. Often we will see i represented in the binary form, i.e. $i \in \{0, 1\}^{\lceil \log i \rceil}$, in such a scenario we simply convert i to its integer representation to index into the string a . To concatenate two strings a and b , we denote it as (a, b) . Lastly, we will consider matrices of the form $A \in \mathbb{F}^{m \times n}$, which we shall view as functions $A : \{0, 1\}^{\lceil \log m \rceil} \times \{0, 1\}^{\lceil \log n \rceil} \mapsto \mathbb{F}$, where $A(i, j)$ corresponds to the element in A along the i -th row, and j -th column.

3.1 Complexity Problems

We define below the two relevant complexity problems, *Boolean circuit satisfiability* (C-SAT) and *satisfiability of systems of rank-1 quadratic equations over a finite field* \mathbb{F} (R1CS). Our starting point will be C-SAT instances, but our protocol will be designed for R1CS instances.

Definition 1 (Circuit-C-SAT). *A circuit satisfiability instance C-SAT is a tuple (C, x) , defined by a Boolean circuit $C : \{0, 1\}^{|x|} \times \{0, 1\}^{|y|} \mapsto \{0, 1\}$ and a string $x \in \{0, 1\}^{|x|}$.*

A C-SAT instance is said to be satisfiable if there exists a string $y \in \{0, 1\}^{|y|}$ such that $C(x, y) = 1$. We denote this as $\mathcal{R}_{\text{C-SAT}}((C, x), y) = 1$.

Definition 2 (R1CS). *An R1CS instance is a tuple $\mathfrak{x} = (\mathbb{F}, A, B, C, io, m, n)$ where (a) io denotes the public input and output of the instance; (b) $A, B, C \in \mathbb{F}^{m \times m}$ with $m \geq |io| + 1$; and (c) there are at most n non-zero entries in each matrix.*

An R1CS instance is said to be satisfiable if there exists a witness $w \in \mathbb{F}^{m - |io| - 1}$ such that $(A \cdot z) \circ (B \cdot z) = (C \cdot z)$, where $z = (io, 1, w)$, \cdot is the matrix-vector product and \circ is the Hadamard (entry-wise) product. We denote this as $\mathcal{R}_{\text{R1CS}}(\mathfrak{x}, w) = 1$.

Circuit-SAT to R1CS. As discussed above, while our definition is for general R1CS instances, we shall consider instances generated via a reduction from C-SAT. Given a C-SAT instance, one can convert it into an R1CS instance over \mathbb{F} where $A, B, C \in \mathbb{F}^{m \times m}$ for $m = O(|C|)$ and $n = O(|C|)$, i.e. the matrices A, B and C are sparse. Furthermore, $io \in \{0, 1\}^{|x|}$ and the witness $w \in \{0, 1\}^{|C| - |x|}$.

We will use the following theorem from [51] that shows that any R1CS instance can be represented by sum over the Boolean hypercube (i.e. over $\{0, 1\}^\ell$ for some ℓ) of a low degree polynomial.

Theorem 2. ([51]). *For any R1CS instance $\mathfrak{x} = (\mathbb{F}, A, B, C, io, m, n)$ there exists a degree 3, $\log m$ -variate polynomial \mathcal{G} such that*

$$\sum_{x \in \{0, 1\}^{\log m}} \mathcal{G}(x) = 0$$

if and only if there exists a witness w such that, except with soundness error negligible in λ , $\mathcal{R}_{\text{R1CS}}(\mathfrak{x}, w) = 1$. Here $|\mathbb{F}|$ is exponential in λ and $m = O(\lambda)$.

4 Somewhere-Extractable Linearly Homomorphic Commitments

In this section, we introduce the notion of somewhere-extractable linearly homomorphic commitment. In such a commitment scheme, one commits to a vector of values using a commitment key that can be generated using one of two indistinguishable modes: *normal mode* or *extraction mode*. Before going into the details, we give an overview of the desired properties from such a scheme:

Somewhere Extraction: When generating the commitment key K in the *extraction mode*, a coordinate i^* is chosen such that any commitment under the key K binds the least significant bit of the i^* -th coordinate of the committed message. Further, alongside the commitment key, the key generation algorithm in the *extraction mode* also outputs a trapdoor td , which allows one to extract the least significant bit of the i^* -th coordinate of the message (i.e. $\text{extraction} \pmod{2}$). We denote this extraction algorithm as $\text{Ext}(\cdot, \text{td})$.

Linear Homomorphism: Consider two commitments $c_1 = \text{Com}(K, \mathbf{m}_1; r_1)$ and $c_2 = \text{Com}(K, \mathbf{m}_2; r_2)$ under the same commitment key (in any mode) for messages $\mathbf{m}_1, \mathbf{m}_2$ with corresponding randomness r_1, r_2 . For any integers a and b , given c_1 and c_2 , there is a way to homomorphically obtain the commitment $\text{Com}(K, a \cdot \mathbf{m}_1 + b \cdot \mathbf{m}_2; a \cdot r_1 + b \cdot r_2)$.

Linearly Homomorphic Extraction: The aforementioned linear homomorphism only concerns well-formed commitments. However, we also need the linear homomorphism properties to hold for commitments that may not be well-formed. To this end, we introduce an extractable space \mathcal{E} , such that for any $c \in \mathcal{E}$, as the name suggests, we can use the extraction algorithm Ext to extract a message. Additionally, we require \mathcal{E} to satisfy the following properties:

Public Verifiability: Given any c , it can be publicly verified if $c \in \mathcal{E}$.

Close Under Linear Homomorphism: The linear homomorphic operation is closed in \mathcal{E} , i.e. for any two elements in \mathcal{E} , the linear homomorphic evaluated commitment is also in \mathcal{E} .

Extraction is Linear Homomorphic: Most importantly, the extraction operation is linear homomorphic in \mathcal{E} , i.e. for any two elements $c_1, c_2 \in \mathcal{E}$, and any two integers a_1, a_2 , we have

$$\text{Ext}(a_1 \cdot c_1 + a_2 \cdot c_2, \text{td}) = a_1 \cdot \text{Ext}(c_1, \text{td}) + a_2 \cdot \text{Ext}(c_2, \text{td}) \pmod{2}$$

Low-Depth Extraction: For our applications, we ideally want the extraction algorithm Ext be computed in low depth, specifically TC^0 . However, this is hard to achieve. Hence, we decompose the extraction algorithm to an offline pre-computation phase PreComp , where PreComp is not allowed to use td but can be of polynomial depth, and a *low-depth* online-phase $\text{OnlineExt}(\cdot, \text{td})$. We only require that the online-phase of extraction $\text{OnlineExt}(\cdot, \text{td})$ be computed in TC^0 .

In Sect. 4.1 we formally define such a commitment scheme. In the full version we also show an extension of the definition to a more general setting, where the commitments are over polynomials. Lastly, in Sect. 4.2, we construct such a commitment scheme from the quadratic residuosity assumption.

4.1 Definition

A somewhere-extractable linearly homomorphic commitment scheme is a tuple of algorithms $\text{LHC} = (\text{Gen}, \text{ExtGen}, \text{Com}, \text{Ext}, \text{Samp})$ described below, where Samp is a randomness sampling algorithm for the commitment.¹⁰

- $\text{Gen}(1^\lambda, 1^k)$: On input the security parameter λ and input length k , outputs a commitment key K .
- $\text{ExtGen}(1^\lambda, 1^k, i^*)$: On input the security parameter λ , input length k and an index $i^* \in [k]$, outputs an *extractable* commitment key K , and a trapdoor td .
- $\text{Com}(K, (x_1, x_2, \dots, x_k); r)$: On input a commitment key K , k integers $(x_1, x_2, \dots, x_k) \in \mathbb{Z}^k$ and randomness $r \leftarrow \text{Samp}(K)$ as input, outputs a commitment c .
- $\text{Ext}(c, \text{td})$: On input a commitment c and a trapdoor td , output a message m . Further, this can be decomposed into two algorithms PreComp and OnlineExt described as follows:
 - $\text{PreComp}(1^\lambda, c)$: On input the security parameter λ and a commitment c , output a *pre-processed* value c' that is to be used for *online extraction*.
 - $\text{OnlineExt}(c', \text{td})$: On input the pre-processed commitment c' and a trapdoor td , output a message $m \in \mathbb{F}_2$.

For correctness, we require that $\text{Ext}(c, \text{td}) = \text{OnlineExt}(\text{PreComp}(1^\lambda, c), \text{td})$. We also emphasize that PreComp does not take the trapdoor as input.

We require the algorithms to satisfy the following properties.

Compactness: The size of the commitment is bounded by some fixed polynomial $\text{poly}(\lambda)$ in the security parameter.

Key Indistinguishability: For any integer $i^* \in [k]$, and any non-uniform PPT adversary \mathcal{D} , there exists a negligible function $\nu(\lambda)$ such that

$$\left| \Pr [K \leftarrow \text{Gen}(1^\lambda, 1^k) : \mathcal{D}(1^\lambda, K) = 1] - \Pr [K \leftarrow \text{ExtGen}(1^\lambda, 1^k, i^*) : \mathcal{D}(1^\lambda, K) = 1] \right| < \nu(\lambda).$$

Linear Homomorphism: There exists a binary operation “+” over the commitments such that, for any key K , $a, b \in \mathbb{Z}$, and any integers vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^k$, and randomness $r, u \in \mathbb{Z}$, we have

$$a \cdot \text{Com}(K, \mathbf{x}; r) + b \cdot \text{Com}(K, \mathbf{y}; u) = \text{Com}(K, a \cdot \mathbf{x} + b \cdot \mathbf{y}; a \cdot r + b \cdot u).$$

¹⁰ We use an explicit randomness sampling algorithm because in our construction from QR, the randomness is sampled from a space that depends on the commitment key.

Extraction: The extraction algorithm Ext satisfies the following properties:

Somewhere \mathbb{F}_2 -Extraction: For any $\mathbf{x} = (x_1, x_2, \dots, x_k) \in \mathbb{Z}^k$, any $i^* \in [k]$, and any randomness $r \in \mathbb{Z}$,

$$\Pr[(K, \text{td}) \leftarrow \text{ExtGen}(1^\lambda, 1^k, i^*), c = \text{Com}(K, \mathbf{x}; r) : \text{Ext}(c', \text{td}) = x_{i^*} \pmod{2}] = 1.$$

Linearly Homomorphic Extraction: There exists an extractable space \mathcal{E} and a polynomial time algorithm $\mathcal{E}\text{Ver}$ such that, for any $c \in \{0, 1\}^*$,

$$\Pr [c \in \mathcal{E} \iff \mathcal{E}\text{Ver}(1^\lambda, c) = 1] = 1.$$

Furthermore, \mathcal{E} is closed under linear combination, i.e. for any $a_1, a_2 \in \mathbb{Z}$, $c_1, c_2 \in \mathcal{E}$, we have $a_1 \cdot c_1 + a_2 \cdot c_2 \in \mathcal{E}$, and

$$\Pr [\text{Ext}(a_1 \cdot c_1 + a_2 \cdot c_2, \text{td}) = a_1 \cdot \text{Ext}(c_1, \text{td}) + a_2 \cdot \text{Ext}(c_2, \text{td}) \pmod{2}] = 1.$$

In addition, every “well-formed” commitment is in \mathcal{E} . i.e. for any key K , input $\mathbf{x} \in \mathbb{Z}^k$, and randomness $r \in \mathbb{Z}$, we have $\text{Com}(K, \mathbf{x}; r) \in \mathcal{E}$.

Low-Depth Online Extraction: The algorithm OnlineExt can be computed by TC^0 circuits.

4.2 Construction

We present our construction of somewhere-extractable linearly homomorphic commitments in Fig. 1.

The reader may note that we have not split the extraction algorithm Ext in Fig. 1 as necessitated by the definition. Instead we defer the decomposition into PreComp and OnlineExt to the full version of the paper. We state below the theorem and defer the proof to the full version of the paper.

Theorem 3. *The construction in Fig. 1 is a somewhere-extractable linearly homomorphic commitment based on the Quadratic Residuosity assumption.*

5 Dual Mode Interactive Batch Arguments for NP

In this section, we define and construct dual mode interactive batch arguments for NP. At a high-level, such an argument system allows for proving multiple instances of an NP language while incurring roughly the communication (and verification) cost of proving a single instance. We consider such protocols in the CRS model that may be executed in one of two modes – *normal* mode or *trapdoor* mode. The former corresponds to normal protocol execution while the latter mode is used in the security proof. Crucially, in the trapdoor mode, we require the protocol to satisfy non-adaptive *statistical* soundness.

Dual Mode Interactive Batch Arguments. We start by providing a formal definition. We shall denote by $\text{Out}_A \langle A(a), B(b) \rangle$ the random variable that corresponds to the output of party A on execution of the protocol between A with input a , and B with input b . Here the probability is taken over the random coins of both A and B .

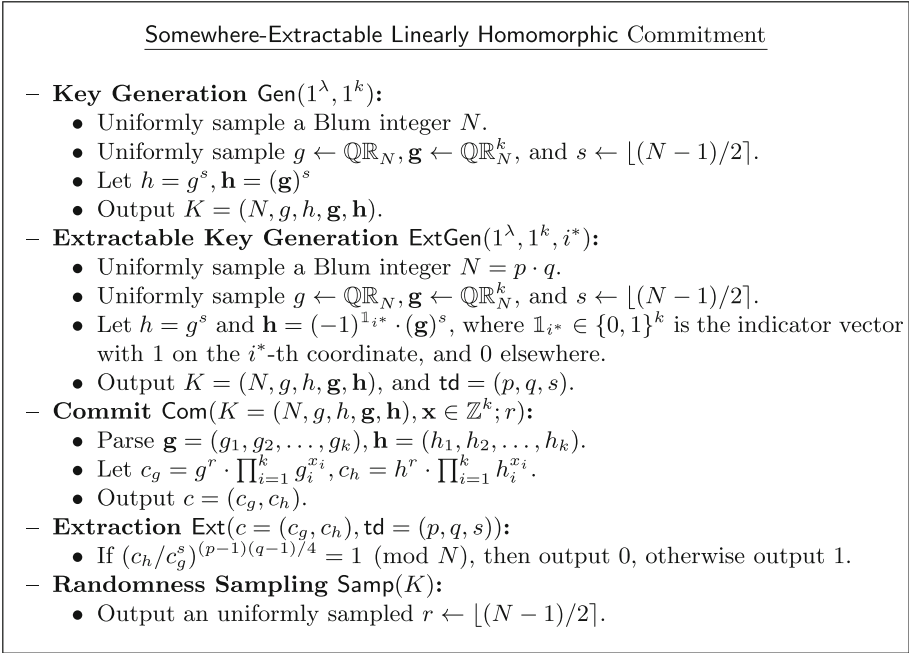


Fig. 1. Construction of somewhere-extractable linearly homomorphic commitment.

Definition 3 (Dual-Mode Interactive Batch Arguments). A dual-mode interactive batch argument, denoted by a tuple of PPT algorithms $(P, V, \text{Gen}, \text{TGen})$, is an interactive protocol in the common reference string (CRS) model for an NP language L defined by relation \mathcal{R}_L if it satisfies the following properties:

Completeness. For all $\mathbf{x} = (x_1, \dots, x_k)$ and $\mathbf{w} = (w_1, \dots, w_k)$ such that for each $i \in [k], \mathcal{R}_L(x_i, w_i) = 1$, it holds that:

$$\Pr [\text{Out}_V \langle P(\text{crs}, \mathbf{x}, \mathbf{w}), V(\text{crs}, \mathbf{x}) \rangle = 1 \mid \text{crs} \leftarrow \text{Gen}(1^\lambda, 1^k)] = 1.$$

Dual Mode Indistinguishability. The two setup modes are computationally indistinguishable, i.e. $\forall k \in \mathbb{N}, \forall i^* \in [k]$,

$$\{\text{crs} : \text{crs} \leftarrow \text{Gen}(1^\lambda, 1^k)\}_{\lambda \in \mathbb{N}} \approx_c \{\text{crs} : \text{crs} \leftarrow \text{TGen}(1^\lambda, 1^k, i^*)\}_{\lambda \in \mathbb{N}}$$

Non-Adaptive Statistical Soundness in Trapdoor Mode. For every (possible unbounded) cheating prover P^* and all $\mathbf{x} = (x_1, \dots, x_k)$ where $\exists i$ s.t. $x_i \notin L$, it holds that $\forall i^* \in [k]$ s.t. $x_{i^*} \notin L$:

$$\Pr [\text{Out}_V \langle P^*(\text{crs}, \mathbf{x}, \mathbf{w}), V(\text{crs}, \mathbf{x}) \rangle = 1 \mid (\text{crs}, \text{td}) \leftarrow \text{TGen}(1^\lambda, 1^k, i^*)] \leq \text{negl}(\lambda).$$

Remark 1. The above definition implies (non-adaptive) soundness against PPT cheating provers in the *normal mode*. This is easily observed via a sequence of

hybrids: (i) switch the crs being generated in the *normal mode* to the *trapdoor mode* for a randomly chosen index i^* while relying on the computational indistinguishability. With probability at least $1/k$ the chosen index i^* will be such that $x_{i^*} \notin L$; (ii) rely on the non-adaptive statistical soundness in the trapdoor mode.

Our Construction. We construct a dual mode interactive batch argument for R1CS, where the instances are generated from instances of Boolean circuit satisfiability that all *share the circuit* C but have different statements x . (We refer the reader to the full version for the corresponding reduction from Boolean satisfiability to R1CS.) This results in k instances of R1CS,

$$\{\mathbf{x}^{(j)}\}_{j \in [k]} = (\mathbb{F}, A, B, C, \{\text{io}^{(j)}\}_{j \in [k]}, m, n),$$

where all the io^j are of the same length, and the instances *share the same* \mathbb{F}, A, B, C, m and n . Furthermore, as a consequence of the reduction, the witnesses to these instances $\{w^{(j)}\}_{j \in [k]}$ are all binary values, i.e. $\forall j \in [k], w^{(j)} \in \{0, 1\}^{m-|\text{io}|-1}$. We work with the field \mathbb{F} , which is an extension field of \mathbb{F}_2 of size 2^λ . Specifically, $\mathbb{F} := \mathbb{F}_2[\alpha]/(v(\alpha))$ for some irreducible polynomial $v(\alpha)$ in \mathbb{F}_2 of degree λ .¹¹

Our protocol relies on a single cryptographic component, namely, a somewhere-extractable linearly homomorphic commitment scheme LHC = (Gen, ExtGen, Com, Ext, Samp) (Sect. 4.1) which can be built (see Sect. 4.2) from the quadratic residuosity assumption. The dual mode property of our protocol comes exclusively from the use of this commitment scheme.

The formal description of the protocol is presented in Fig. 2.

Below, we provide an overview of our protocol, focusing on how we implement batching. We note that due to the lack of space, we omit some details regarding the underlying polynomials used in our construction, and refer the reader to the full version for the details.

1. Given k R1CS instances $\{\mathbf{x}^{(j)}\}_{j \in [k]}$, the prover needs to commit to k witnesses $\{w^{(j)}\}_{j \in [k]}$, where the witnesses are all of the same size $|w|$. This is done by first representing the k witnesses as a matrix W , where each witness occupies a single row. The prover uses LHC to commit to $|w|$ k -tuples corresponding to each column of the matrix W . From the compactness property of LHC, the total size of the commitments sent by the prover is proportional to the size of a *single witness* $|w|$.

An observant reader may note that the message space for a k -tuple commitment in LHC is \mathbb{Z}^k , and not \mathbb{F}^k as we would like. Here we utilize the fact that the prover is committing to the witness whose values are only binary (as consequence of the reduction from C-SAT to R1CS, see Sect. 3.1)¹², and

¹¹ One can think of the representation to be a λ length vector in \mathbb{F}_2 corresponding to the coefficients of the polynomial $f \in \mathbb{F}_2[\alpha]/(v(\alpha))$.

¹² Our protocol does not handle arbitrary R1CS instances where the witness may have values in \mathbb{F} outside of $\{0, 1\}$.

Protocol: Interactive Batch Argument (P, V, Gen, TGen)**Common Reference String:** $K \leftarrow \text{LHC.Setup}_1(1^\lambda, 1^k)$ **Common input:** input $\{\mathbb{x}^{(j)}\}_{j \in [k]} := (\mathbb{F}, A, B, C, \{\text{io}^{(j)}\}_{j \in [k]}, m, n)$, security parameter 1^λ **P's auxiliary input:** witnesses $\{w^{(j)}\}_{j \in [k]}$ such that $\forall j \in [k], \mathcal{R}_{\text{RICS}}(\mathbb{x}^{(j)}, w^{(j)}) = 1$

1. Prover commits to the k -tuple of witnesses in a *column-wise* fashion. Specifically $\forall y \in \{0, 1\}^{s_2}$, $c_y := \text{LHC.Com}(K, (w_y^{(1)}, \dots, w_y^{(k)}); r_y)$ where $r_y \leftarrow_s \text{LHC.Samp}(K)$. Send $\{c_y\}_{y \in \{0, 1\}^{s_2}}$ to the verifier V.
2. Verifier V samples a random vector $\tau \leftarrow_s \mathbb{F}^s$ and sends τ to the prover P.
3. Prover P and verifier V run k sumcheck protocols ($\text{P}_{\text{SC}}(\mathcal{G}_{\text{io}^{(j)}}, \tau), \text{V}_{\text{SC}}(0)$)¹² in parallel where the verifier uses the same randomness in each sumcheck. The output of the sumchecks are $r^* \in \mathbb{F}^s$ and $\{\nu^{(j)}\}_{j \in [k]}$.
4. The verifier V asks P for values $\{\nu_{A,2}^{(j)}, \nu_{B,2}^{(j)}, \nu_{C,2}^{(j)}\}_{j \in [k]}$.
5. Prover P computes $\forall j \in [k], X \in \{A, B, C\}$,
 - $\nu_{X,2}^{(j)} := \sum_{y \in \{0, 1\}^{s_2}} \tilde{X}''(r^*, y) \cdot w_y^{(j)}$,
 - $r_X := \sum_{y \in \{0, 1\}^{s_2}} \tilde{X}''(r^*, y) \cdot r_y$,
 sends $\{\nu_{A,2}^{(j)}, \nu_{B,2}^{(j)}, \nu_{C,2}^{(j)}\}_{j \in [k]}$ along with the *commitment openings* r_A, r_B, r_C to the verifier V.
6. The verifier V does the following
 - (a) computes $\forall X \in \{A, B, C\}$, $c_X := \sum_{y \in \{0, 1\}^{s_2}} \tilde{X}''(r^*, y) \cdot c_y$
 - (b) checks commitment opening, $\forall X \in \{A, B, C\}$, $c_X \stackrel{?}{=} \text{LHC.Com}(K, (\nu_{X,2}^{(1)}, \dots, \nu_{X,2}^{(k)}); r_X)$
 - (c) computes $\forall j \in [k], X \in \{A, B, C\}$, $\nu_{X,1}^{(j)} := \sum_{y \in \{0, 1\}^{s_1}} \tilde{X}'(r^*, y) \cdot (\text{io}||1)_y^{(j)}$, and $\nu_X^{(j)} = \nu_{X,1}^{(j)} + \nu_{X,2}^{(j)}$.
 - (d) checks $\forall j \in [k]$, $\nu^{(j)} = (\nu_A^{(j)} \cdot \nu_B^{(j)} - \nu_C^{(j)}) \cdot \tilde{\text{eq}}(r^*, \tau)$, and reject if any of the checks fail.
 Accept if none of the checks have failed.

Fig. 2. Interactive Batch Argument for RICS.

therefore the message space for the commitment of a k -tuple is $\{0, 1\}^k \subset \mathbb{Z}^k$. This also explains why we commit to the witness w rather than its multilinear extension, since they are equivalent when the witness is a binary string.

Let $\{c_y\}_{y \in [|w|]}$ denote the commitments.

2. The prover and verifier run sumcheck protocols for polynomials $\{\mathcal{G}_{\tau, \text{io}}^{(j)}\}_{j \in [k]}$ - there are k distinct polynomials since the witness (which determines the polynomial) for each RICS instance is different. Specifically, the sumcheck protocol is to prove that the following sum

$$\sum_{x \in \{0, 1\}^{\log m}} \mathcal{G}_{\tau, \text{io}}^{(j)} := \sum_{x \in \{0, 1\}^{\log m}} \tilde{F}_{\text{io}}(x) \cdot \tilde{\text{eq}}(x, \tau)$$

is 0, where \tilde{F}_{io} is a polynomial that depends on A, B, C, io and w . The prover and verifier run all k sumchecks in parallel, where the verifier uses the *same randomness* across all the sumcheck protocols.

- At the end of the sumcheck protocol, the verifier needs to evaluate each polynomial $\{\mathcal{G}_{\tau, \text{io}}^{(j)}\}_{j \in [k]}$ at a point $r^* \in \mathbb{F}^{\log m}$ determined by the sumcheck polynomial. Note that since the verifier used the same randomness across all the sumcheck protocols, it is the same value r^* for *all* the polynomials $\{\mathcal{G}^{(j)}\}_{j \in [k]}$. Since $\tilde{\text{eq}}(r^*, \tau)$ can be computed locally by the verifier, the check at the end of the sumcheck reduces to computing, for each $j \in [k]$,

$$\begin{aligned} \tilde{F}_{\text{io}}^{(j)}(r^*) := & \left(\sum_{y \in \{0,1\}^{s_1}} \tilde{A}'(r^*, y) \cdot (\text{io}^{(j)}, 1)_y + \sum_{y \in \{0,1\}^{s_2}} \tilde{A}''(r^*, y) \cdot w_y^{(j)} \right) \\ & \left(\sum_{y \in \{0,1\}^{s_1}} \tilde{B}'(r^*, y) \cdot (\text{io}^{(j)}, 1)_y + \sum_{y \in \{0,1\}^{s_2}} \tilde{B}''(r^*, y) \cdot w_y^{(j)} \right) \\ & - \left(\sum_{y \in \{0,1\}^{s_1}} \tilde{C}'(r^*, y) \cdot (\text{io}^{(j)}, 1)_y + \sum_{y \in \{0,1\}^{s_2}} \tilde{C}''(r^*, y) \cdot w_y^{(j)} \right) \end{aligned}$$

The terms that depend solely on the instance and common input can be computed by the verifier locally. The remaining terms that depend on the witness, highlighted in red above, are terms that the prover needs to send to the verifier. We denote these terms by $\{\nu_{A,2}^{(j)}, \nu_{B,2}^{(j)}, \nu_{C,2}^{(j)}\}_{j \in [k]}$, where for each $j \in [k]$,

$$\begin{aligned} \nu_{A,2}^{(j)} &:= \sum_{y \in \{0,1\}^{s_2}} \tilde{A}''(r^*, y) \cdot w_y^{(j)}, & \nu_{B,2}^{(j)} &:= \sum_{y \in \{0,1\}^{s_2}} \tilde{B}''(r^*, y) \cdot w_y^{(j)} \\ \nu_{C,2}^{(j)} &:= \sum_{y \in \{0,1\}^{s_2}} \tilde{C}''(r^*, y) \cdot w_y^{(j)} \end{aligned}$$

There are **two crucial observations** to be made here: (i) for each fixed r^* , the above values are simply a linear combination of each individual witness $w^{(j)}$ (with appropriate coefficients); and (ii) the linear coefficients are the *same for each witness*, and in fact the linear coefficients depend only on the index of the witness.

- The above observations allow the prover to open the commitments to $\{\nu_{A,2}^{(j)}, \nu_{B,2}^{(j)}, \nu_{C,2}^{(j)}\}_{j \in [k]}$ by the linear homomorphism property of LHC since the property allows for the same linear coefficient to be applied *all* values in the k -tuple within the commitment.

Specifically, the prover sends the values (and randomness) in the clear to the verifier, who then performs the same linear homomorphism over the committed values $\{c_y\}_{y \in [|w|]}$ to check if the openings sent by the prover are correct before computing the checks necessitated by the sumcheck.

While we remarked that the values that are inside the commitment are binary, this is not true of the linear coefficients that are in \mathbb{F} . But since $\mathbb{F} = \mathbb{F}_2[\alpha]/(v(\alpha))$, by the *homomorphism with respect to polynomial* property of LHC and that $\mathbb{F}_2[\alpha]/(v(\alpha)) \subset \mathbb{Z}[\alpha]/(v(\alpha))$, it is fine that coefficients are in \mathbb{F} .

Lastly it should be noted that the homomorphism properties are defined over $\mathbb{Z}[\alpha]/(v(\alpha))$ and not $\mathbb{F}_2[\alpha]/(v(\alpha))$, meaning there is no modular reduction in the coefficients of the resultant polynomial after the homomorphism operation, i.e. the λ length vector has elements in \mathbb{Z} instead of \mathbb{F}_2 . Therefore, to verify correctness of the commitment, the prover *computes the values* $\{\nu_{A,2}^{(j)}, \nu_{B,2}^{(j)}, \nu_{C,2}^{(j)}\}_{j \in [k]}$ over $\mathbb{Z}[\alpha]/(v(\alpha))$, i.e. no modular reduction. The verifier does the commitment check over $\mathbb{Z}[\alpha]/(v(\alpha))$, but once the check passes successfully reduces to $\mathbb{F}_2[\alpha]/(v(\alpha))$ ¹³. While this increases the number of bits sent by the prover, we show in the full version of the paper that the increase is quite small, giving us the following theorem.

Theorem 4. *Assuming the existence of somewhere-extractable linearly homomorphic commitments, the protocol in Fig. 2 is a dual-mode interactive batch argument for R1CS where*

- Total communication cost is $O(m\lambda + (\lambda + k) \log m)$
- The verifier’s total run time is $O(k|io| + n + m) \cdot \text{poly}(\lambda)$

where all instances have the same length $|io|$.

We defer the proof to the full version of the paper. Since we start with Boolean circuit satisfiability to generate R1CS instances, we get the following corollary with costs corresponding to the size the Boolean circuit.

Corollary 1. *If we start with C-SAT instances defined by a boolean circuit $C : \{0, 1\}^{|x|} \times \{0, 1\}^{|y|} \mapsto \{0, 1\}$, then the protocol in Fig. 2 is a dual-mode interactive batch argument for C-SAT where*

- Total communication cost is $O(|C| + k \log |C|) \text{poly}(\lambda)$
- The verifier’s total run time is $O(k|x| + |C|) \cdot \text{poly}(\lambda)$

6 Non-interactive Batch Arguments for NP

We now construct a non-interactive batch argument system for NP. We start by formally defining this notion below.

Definition 4 (Non-interactive Batch Arguments). *A non-interactive batch argument for an NP language L defined by relation \mathcal{R}_L is a tuple of algorithms $(\text{Gen}, \text{P}, \text{V})$ satisfying the following properties:*

¹³ This is just reducing each element in the λ length vector to \mathbb{F}_2 .

- **Completeness:** For all $\mathbf{x} = (x_1, \dots, x_k)$ and $\mathbf{w} = (w_1, \dots, w_k)$ such that for each $i \in [k]$, $\mathcal{R}_L(x_i, w_i) = 1$, it holds that:

$$\Pr[\mathbf{V}(\text{crs}, \mathbf{x}, \pi) = 1 \mid \text{crs} \leftarrow \text{Gen}(1^\lambda), \pi \leftarrow \mathbf{P}(\text{crs}, \mathbf{x}, \mathbf{w})] = 1.$$

- **(Non-adaptive) Soundness:** For every PPT adversary \mathbf{P}^* and all $\mathbf{x} = (x_1, \dots, x_k)$ where $\exists i$ s.t. $x_i \notin L$, it holds that:

$$\Pr[\mathbf{V}(\text{crs}, \mathbf{x}, \pi) = 1 \mid \text{crs} \leftarrow \text{Gen}(1^\lambda), \pi \leftarrow \mathbf{P}^*(\text{crs})] \leq \text{negl}(\lambda).$$

In the full version, we show that the Fiat-Shamir transform w.r.t. \mathcal{H} when applied to any strongly FS compatible protocol is sound as long as \mathcal{H} is correlation intractable for TC^0 . Next, we construct a non-interactive batch argument system for NP by demonstrating that the above transformation remains sound when applied to our dual-mode interactive batch argument (Sect. 5), even though the aforementioned protocol does not satisfy strong FS compatibility. Finally, we show that although our described protocol has a linear dependence on k , this can be made sub-linear.

Acknowledgments. Arka Rai Choudhuri, Abhishek Jain and Zhengzhong Jin are supported in part by NSF CNS-1814919, NSF CAREER 1942789 and Johns Hopkins University Catalyst award. Arka Rai Choudhuri and Abhishek Jain are also supported in part by the Office of Naval Research Grant N00014-19-1-2294. Arka Rai Choudhuri is also supported in part by NSF Grant CNS-1908181. Zhengzhong Jin is also supported in part by NSF CAREER 1845349.

References

1. Barak, B.: How to go beyond the black-box simulation barrier. In: 42nd FOCS, pp. 106–115. IEEE Computer Society Press, October 2001
2. Bartusek, J., Bronfman, L., Holmgren, J., Ma, F., Rothblum, R.D.: On the (in)security of Kilian-based SNARGs. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 522–551. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_20
3. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) ITCS 2012, pp. 326–349. ACM, January 2012
4. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 111–120. ACM Press, June 2013
5. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic Residuosity Strikes Back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_1
6. Brakerski, Z., Holmgren, J., Kalai, Y.T.: Non-interactive delegation and batch NP verification from standard computational assumptions. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC, pp. 474–482. ACM Press, June 2017

7. Brakerski, Z., Koppula, V., Mour, T.: NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 738–767. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_26
8. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 535–564. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_20
9. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. *J. Cryptol.* **25**(4), 723–747 (2012)
10. Canetti, R., et al.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC, pp. 1082–1090. ACM Press, June 2019
11. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 91–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_4
12. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. *J. ACM* **51**(4), 557–594 (2004)
13. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: 36th FOCS, pp. 41–50. IEEE Computer Society Press, October 1995
14. Choudhuri, A.R., Hubáček, P., Kamath, C., Pietrzak, K., Rosen, A., Rothblum, G.N.: Finding a Nash equilibrium is no easier than breaking Fiat-Shamir. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC, pp. 1103–1114. ACM Press, June 2019
15. Choudhuri, A.R., Jain, A., Jin, Z.: Non-interactive batch arguments for np from standard assumptions. Cryptology ePrint Archive, Report 2021/807 (2021). <https://eprint.iacr.org/2021/807>
16. Ciampi, M., Parisella, R., Venturi, D.: On adaptive security of delayed-input sigma protocols and Fiat-Shamir NIZKs. In: Galdi, C., Kolesnikov, V. (eds.) SCN 2020. LNCS, vol. 12238, pp. 670–690. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57990-6_33
17. Couteau, G., Katsumata, S., Ursu, B.: Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 442–471. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_15
18. Damgård, I., Faust, S., Hazay, C.: Secure two-party computation with low communication. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 54–74. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_4
19. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18
20. Döttling, N., Garg, S., Hajiabadi, M., Masny, D.: New constructions of identity-based and key-dependent message secure encryption schemes. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 3–31. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_1
21. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 3–32. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_1

22. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
23. Garg, S., Hajiabadi, M.: Trapdoor functions from the computational Diffie-Hellman assumption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 362–391. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_13
24. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC, pp. 99–108. ACM Press, June 2011
25. Goldreich, O., Håstad, J.: On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.* **67**(4), 205–214 (1998)
26. Goldreich, O., Vadhan, S.P., Wigderson, A.: On interactive proofs with a laconic prover. *Comput. Complex.* **11**(1–2), 1–53 (2002)
27. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS, pp. 102–115. IEEE Computer Society Press, October 2003
28. Goldwasser, S., Lin, H., Rubinfeld, A.: Delegation of computation without rejection problem from designated verifier CS-Proofs. *Cryptology ePrint Archive*, Report 2011/456 (2011). <http://eprint.iacr.org/2011/456>
29. Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In: Thorup, M. (ed.) 59th FOCS, pp. 850–858. IEEE Computer Society Press, October 2018
30. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) ITCS 2015, pp. 163–172. ACM, January 2015
31. Jain, A., Jin, Z.: Non-interactive zero knowledge from sub-exponential DDH. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 3–32. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_1
32. Jawale, R., Kalai, Y.T., Khurana, D., Zhang, R.: SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In: STOC. ACM (2021)
33. Jawale, R., Khurana, D.: Lossy correlation intractability and PPAD hardness from sub-exponential LWE. *Cryptology ePrint Archive*, Report 2020/911 (2020). <https://eprint.iacr.org/2020/911>
34. Kalai, Y.T., Paneth, O., Yang, L.: How to delegate computations publicly. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC, pp. 1115–1124. ACM Press, June 2019
35. Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 224–251. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_8
36. Kalai, Y.T., Zhang, R.: SNARGs for bounded depth computations from sub-exponential LWE. *Cryptology ePrint Archive*, Report 2020/860 (2020). <https://eprint.iacr.org/2020/860>
37. Kaslasi, I., Rothblum, G.N., Rothblum, R.D., Sealfon, A., Vasudevan, P.N.: Batch verification for statistical zero knowledge proofs. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 139–167. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_6

38. Kaslasi, I., Rothblum, R.D., Vasudevanr, P.N.: Public-coin statistical zero-knowledge batch verification against malicious verifiers. In: Canteaut, A., Stan- daert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 219–246. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_8
39. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC, pp. 723–732. ACM Press, May 1992
40. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, pp. 419–428. ACM Press, June 2015
41. Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: 38th FOCS, pp. 364–373. IEEE Computer Society Press, October 1997
42. Lombardi, A., Vaikuntanathan, V.: Fiat-Shamir for repeated squaring with appli- cations to PPAD-hardness and VDFs. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 632–651. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_22
43. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. *J. ACM* **39**(4), 859–868 (1992)
44. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS, pp. 436–453. IEEE Computer Society Press, November 1994
45. Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_6
46. Okamoto, T., Pietrzak, K., Waters, B., Wichs, D.: New realizations of somewhere statistically binding hashing and positional accumulators. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 121–145. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_6
47. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (Plain) learn- ing with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 89–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_4
48. Reingold, O., Rothblum, G.N., Rothblum, R.D.: Constant-round interactive proofs for delegating computation. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC, pp. 49–62. ACM Press, June 2016
49. Reingold, O., Rothblum, G.N., Rothblum, R.D.: Efficient batch verification for UP. In: Computational Complexity Conference. LIPIcs, vol. 102, pp. 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018)
50. Rothblum, G.N., Rothblum, R.D.: Batch verification and proofs of proximity with polylog overhead. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 108–138. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_5
51. Setty, S.: Spartan: efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_25
52. Shamir, A.: $IP = PSPACE$. *J. ACM* **39**(4), 869–877 (1992)



Targeted Lossy Functions and Applications

Willy Quach¹(✉), Brent Waters^{2,3}, and Daniel Wichs^{1,3}

¹ Northeastern University, Boston, USA
quach.w@northeastern.edu, wichs@ccs.neu.edu

² UT Austin, Austin, USA
bwaters@cs.utexas.edu

³ NTT Research, Sunnyvale, USA

Abstract. Lossy trapdoor functions, introduced by Peikert and Waters (STOC '08), can be initialized in one of two indistinguishable modes: in injective mode, the function preserves all information about its input, and can be efficiently inverted given a trapdoor, while in lossy mode, the function loses some information about its input. Such functions have found countless applications in cryptography, and can be constructed from a variety of Cryptomania assumptions. In this work, we introduce *targeted lossy functions (TLFs)*, which relax lossy trapdoor functions along two orthogonal dimensions. Firstly, they do not require an inversion trapdoor in injective mode. Secondly, the lossy mode of the function is initialized with some target input, and the function is only required to lose information about this particular target. The injective and lossy modes should be indistinguishable even given the target. We construct TLFs from Minicrypt assumptions, namely, injective pseudorandom generators, or even one-way functions under a natural relaxation of injectivity. We then generalize TLFs to incorporate *branches*, and construct *all-injective-but-one* and *all-lossy-but-one* variants. We show a wide variety of applications of targeted lossy functions. In several cases, we get the first Minicrypt constructions of primitives that were previously only known under Cryptomania assumptions. Our applications include:

- *Pseudo-entropy functions* from one-way functions.
- Deterministic leakage-resilient message-authentication codes and improved leakage-resilient symmetric-key encryption from one-way functions.
- Extractors for *extractor-dependent sources* from one-way functions.
- Selective-opening secure symmetric-key encryption from one-way functions.
- A new construction of CCA PKE from (exponentially secure) trapdoor functions and injective pseudorandom generators.

We also discuss a fascinating connection to distributed point functions.

B. Waters—Supported by NSF CNS-1908611, Packard Foundation Fellowship, and Simons Investigator Award.

D. Wichs—Research supported by NSF grant CNS-1750795 and the Alfred P. Sloan Research Fellowship.

1 Introduction

Lossy trapdoor functions, introduced by Peikert and Waters [PW08], are a fundamental cryptographic tool and have found countless applications in many areas of cryptography. They can be constructed from a wide variety of specific number-theoretic and algebraic “Cryptomania” assumptions. In this work, we introduce a relaxation of lossy trapdoor functions that we call *targeted-lossy functions* (TLFs), and show how to instantiate them using “Minicrypt” assumptions, such as injective pseudorandom generators or even one-way functions for a natural variant. We then provide applications of TLFs to a diverse set of problems. For several of these problems, we get the first solutions under one-way functions, where previously only solutions under specific Cryptomania assumptions were known.

Lossy Trapdoor Functions. Lossy trapdoor functions consist of a function family $F_{\text{fk}}(\cdot)$ indexed by a public *function key* fk . The function key fk can be generated in one of two modes. In *injective* mode, the function $F_{\text{fk}}(\cdot)$ is injective, and therefore each output $y = F_{\text{fk}}(x)$ uniquely determines the input x . Furthermore, the public function key fk is generated together with a secret trapdoor td that allows one to efficiently invert the function and recover the input x from the output $y = F_{\text{fk}}(x)$. In *lossy* mode, the output $y = F_{\text{fk}}(x)$ loses some information about the input x . This is captured by defining a lossiness parameter ℓ and requiring that the size of the image of F_{fk} is at most a $\frac{1}{2^\ell}$ fraction of the size of the domain. In particular, this implies that when x is uniformly random over its domain, then the conditional entropy¹ of x given $F_{\text{fk}}(x)$ is at least ℓ bits, meaning that this information about x is lost by $F_{\text{fk}}(x)$. The two modes should be *computationally indistinguishable*: given fk , one cannot tell if it was generated in injective mode or lossy mode.

Since their introduction, lossy trapdoor functions have turned out to be incredibly versatile tool and have quickly become an integral part of our cryptographic tool-set. They have found countless and varied applications, including to CCA security, trapdoor functions with many hard-core bits, collision-resistant hash functions, and oblivious transfer [PW08], deterministic encryption [BFO08], analyzing OAEP [KOS10], hedged public-key encryption with bad randomness [BBN+09], selective opening security [BHY09], pseudo-entropy functions [BHK11], point-function obfuscation [Zha16], computational extractors [DVW20, GKK20], incompressible encodings [MW20], etc.

Lossy trapdoor functions are known to imply public-key encryption, making them a *Cryptomania* primitive. We currently know how to construct lossy trapdoor functions under most (but not all) concrete Cryptomania assumptions, such as DDH, LWE, Quadratic Residuosity (QR), Decision-Composite Residuosity (DCR), and Phi-hiding [PW08, KOS10, FGK+13], but not e.g., factoring, RSA, or the (low noise) LPN assumption.

¹ Throughout the introduction, entropy refers to min-entropy, and conditional entropy refers to average-case conditional min-entropy [DORS08].

Targeted-Lossy Functions. In this work, we introduce a relaxation of lossy trapdoor functions, that we call *targeted-lossy functions (TLFs)*, with the goal of constructing them under weaker assumptions. TLFs relax the notion of lossy trapdoor functions along two orthogonal dimensions:

- *No inversion trapdoor in injective mode.* When we generate fk in injective mode, we now *only* require that the function $F_{\text{fk}}(\cdot)$ is injective, but we no longer require there to be a trapdoor td that allows us to efficiently invert it.
- *Targeted Lossiness.* When we generate fk in lossy mode, we are now also given a target input x^* and only require that $F_{\text{fk}}(x^*)$ loses ℓ bits of information about the particular target x^* . In particular, when the target x^* is chosen uniformly at random and fk is chosen in lossy mode for this target, then the conditional entropy of x^* given $(\text{fk}, F_{\text{fk}}(x^*))$ should be at least ℓ bits.

The two modes should be computationally indistinguishable even given the potential target x^* . In other words, given the pair (fk, x^*) , one cannot distinguish whether fk was chosen in injective mode and independently of x^* or in lossy mode with x^* as the target.

Notice that the first relaxation already appears to take us out of Cryptomania – without a trapdoor, there is no obvious way to use this primitive to construct public-key encryption. This relaxation was considered on its own in prior works (e.g., [BHK11, DVW20]), and is already known to have interesting applications. Unfortunately, there has been no progress towards achieving this relaxation on its own under any Minicrypt assumption, or even under any assumption that doesn't already imply the full notion of lossy trapdoor functions.² This motivates us to consider this relaxations in conjunction with our second relaxation.

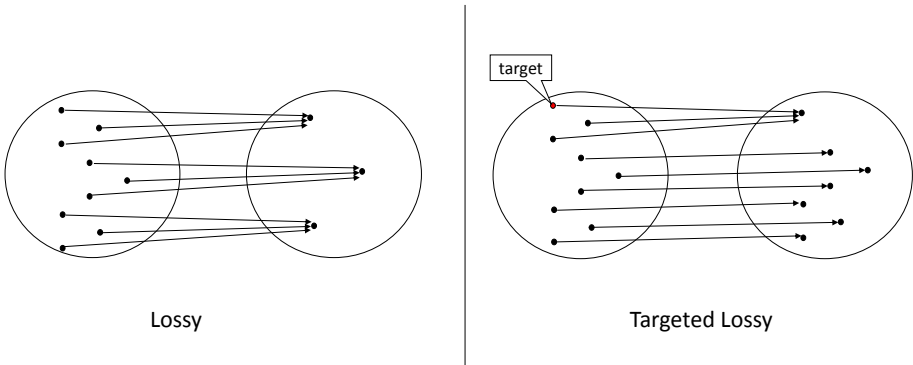


Fig. 1. Lossy vs Targeted-Lossy

² It is also known that, with a sufficiently high lossiness rate, this relaxation on its own would already at least imply collision-resistant hashing [PW08], and therefore is unlikely to follow from one-way functions/permutations.

The second relaxation substantially weakens the lossiness requirement and only asks for targeted lossiness. To highlight the difference between standard vs targeted lossiness, notice that targeted lossiness with parameter ℓ could be achieved by choosing a lossy function key fk for some target x^* , where the output $y = F_{\text{fk}}(x^*)$ has 2^ℓ pre-images in the set $S = \{x : F_{\text{fk}}(x) = y\}$, but for every $x \notin S$, the value $F_{\text{fk}}(x)$ has a unique pre-image x . Such a function would *not* be lossy in the standard sense. For example, if the domain of the function is $\{0, 1\}^n$ with $n = 2\ell$, then, from the point of view of standard lossiness, the function only has negligible information loss $\ell' = O(2^{-\ell})$, even though its targeted-lossiness ℓ can be an arbitrarily large polynomial.³ This example is illustrated in Fig. 1. Despite the significant difference between the notions, we show that targeted lossiness suffices in many applications in place of standard lossiness.⁴

TLFs with Branches/Tags. We also consider an augmented notion of TLFs with branches/tags, analogously to prior notions of branches for lossy trapdoor functions [PW08]. In this setting, a single function key fk defines an entire family of functions $F_{\text{fk}, \text{tag}}(\cdot)$ with various branches indexed by tag . We can sample the function key fk with a special branch tag^* and a target value x^* , and we require that the pair (fk, x^*) computationally hides tag^* . We define two main variants of this notion, depending on whether the special branch is lossy or injective.

In a targeted *all-injective-but-one* (T-AIBO) family, the special branch tag^* is targeted-lossy and all other branches are injective. In particular, for all $\text{tag} \neq \text{tag}^*$, the function $F_{\text{fk}, \text{tag}}$ is injective, while $F_{\text{fk}, \text{tag}^*}(x^*)$ loses ℓ bits of information about the target x^* . This notion is most directly analogous to the way branches were defined for standard lossy trapdoor functions of [PW08].

In a targeted *all-lossy-but-one* (T-ALBO) family, the function $F_{\text{fk}, \text{tag}^*}$ is injective on the special branch tag^* , while all other branches $F_{\text{fk}, \text{tag}}$ are *cumulatively* targeted-lossy. In particular, the cumulative outputs of all lossy branches $(F_{\text{fk}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}$ must lose ℓ -bits of information about the target x^* . An analogous notion of branches for the case of lossy trapdoor functions was previously considered in [CPW20], and a relaxed version without trapdoors (but without the second relaxation to targeted lossiness) was considered implicitly in [BHK11, GKK20] and explicitly in [DVW20]. All prior constructions relied on Cryptomania assumptions.

Relaxing Injectivity. It turns out that we can also relax the injectivity requirement of TLFs, while sufficing for most of our applications. When we choose fk in injective mode, instead of requiring that $F_{\text{fk}}(x)$ uniquely determines x , we

³ The function has an image of size $2^n - 2^\ell + 1$, which we can write as $\frac{1}{2^{\ell'}} 2^n$ for $\ell' = O(2^{-\ell})$.

⁴ We could also consider the second relaxation to targeted lossiness on its own, without making the first relaxation (i.e., by still insisting on an inversion trapdoor in injective mode). In that case, the resulting notion would still be a Cryptomania primitive. Interestingly, this notion was considered informally in [GGH19], where it was constructed under the CDH assumption, which is not known to imply standard lossy trapdoor functions. Not much else is known about this setting.

only require that it uniquely determines some *property* of x , modeled as an arbitrary function $P(x)$. In this case, we also require that when fk is in lossy mode for the target value x^* , then $F_{\text{fk}}(x^*)$ loses ℓ bits of information about the same property $P(x^*)$. We can define T-AIBOs and T-ALBOs with relaxed injectivity analogously, and even allow the property P to depend on fk . In the case of T-ALBOs, by setting the property $P(x) = F_{\text{fk}, \text{tag}^*}(x)$, this relaxed injectivity requirement is equivalent to insisting that the cumulative outputs of all lossy branches $(F_{\text{fk}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}$ should lose ℓ bits of information about the output of the “injective” branch $F_{\text{fk}, \text{tag}^*}(x^*)$.

1.1 Our Results

We construct targeted lossy functions (TLFs) from injective pseudorandom generators (PRGs). We also generalize our construction to targeted all-injective-but-one functions (T-AIBOs) under the same assumption. For all-lossy-but-one functions (T-ALBOs), we need a stronger “doubly injective” PRG $G(x) = (y_0, y_1)$, whose output consists of two halves y_0, y_1 , and the PRG is injective on each half individually (i.e., either one of y_0, y_1 uniquely determines x). We also construct TLFs, T-AIBOs and T-ALBOs with relaxed injectivity from just one-way functions. In all cases, we start with a basic construction that only achieves lossiness of $\ell = 1$ bits, but can then amplify lossiness via parallel repetition to get to an arbitrary polynomial ℓ . (However, the lossiness *rate* of our constructions, defined as ℓ/n where n is the domain size, is only $1/\lambda$, where λ is the security parameter. Achieving a higher lossiness rate in Minicrypt is a fascinating open problem.)

Application: Pseudo-entropy Functions. The work of Braverman, Hassidim, and Kalai [BHK11] introduced the notion of a *pseudo-entropy function (PEF)*, and constructed them under the DDH assumption. A PEF $f_k(x)$ has a secret key k and takes as inputs values x . The requirement is that for any a-priori chosen input x^* , we can indistinguishably select the key k so that $f_k(x^*)$ has ℓ bits of true statistical entropy even given $f_k(x)$ for all inputs $x \neq x^*$. We observe that PEFs follow almost immediately from T-ALBOs with relaxed injectivity (just by “renaming” the various components), and therefore get a construction of PEFs from one-way functions. The amount of entropy ℓ in our construction can be set to an arbitrarily large polynomial. (On the other hand, the entropy *rate* of our construction, defined as ℓ/n where n is the key size, is stuck at $1/\lambda$. This is in contrast to the construction of [BHK11], which achieved an entropy rate of $1 - o(1)$ under DDH.)

Application: Leakage-Resilience. *Leakage-resilient cryptography* aims to preserve security even if an adversary can get some partial leakage on the secret key. We consider the setting of memory leakage [AGV09, ADW09, NS09, HLWW13], where an adversary can learn any efficiently computable function of the secret key, as long as the number of leaked bits is bounded by some parameter ℓ . As shown by [BHK11], pseudo-entropy functions (PEFs) are useful for leakage-resilient symmetric-key cryptography and were previously used to construct

(selectively secure) deterministic leakage-resilient MACs under DDH. By using our new construction of PEFs from one-way functions, we get the first (selectively secure) deterministic leakage-resilient MACs from just one-way functions. The amount of leakage ℓ that we can tolerate can be set to an arbitrarily large polynomial. (However, the length of the key grows depending on ℓ and the leakage *rate* of our construction, defined as ℓ/n where n is the key size, is stuck at $1/\lambda$. This is in contrast to the construction of [BHK11], which achieved a leakage rate of $1 - o(1)$ under DDH.) We can also use a similar technique to construct leakage-resilient CPA-secure symmetric-key encryption from one-way functions.

We note that a prior work of [HLWW13] constructed leakage-resilient symmetric-key primitives, including CPA-secure symmetric-key encryption and (adaptively secure) MACs from one-way functions. The amount of leakage and the leakage rate are the same as in our construction. However, the MACs in the prior work were inherently randomized, while in this work we get deterministic MACs. This is especially crucial in the context of leakage-resilience since, in a randomized construction, leakage that occurs during a computation may also depend on the randomness of the computation in addition to the secret key, but such leakage was not analyzed by the prior work (and indeed, the proof there would fail). Furthermore, our MAC has a smaller signature size: the ratio of leakage to signature size is $(1 - o(1))$ in our construction while it is $1/\lambda$ in the prior work. For the case of CPA-secure symmetric-key encryption, in the prior work the ciphertext size grew linearly with the leakage bound ℓ , while in our work, only the secret key size grows with the leakage bound ℓ , but the ciphertext size just has a minimal $O(\lambda)$ additive overhead on top of the message length. For both MACs and symmetric-key encryption, our constructions are substantially different from those of [HLWW13].

Application: Extractor-Dependent Sources. The work of Dodis, Vaikuntanathan and Wichs [DVW20], which we will refer to as DVW, defined the notion of (computational) extractors for extractor-dependent sources. The goal is to extract nearly uniform randomness R from an arbitrary source of randomness X that has some sufficient entropy. Classical results show this to be possible using a *seeded* randomness extractor $R = \text{Ext}(X; S)$, which relies on a public random seed S . As long as the source X is independent of the seed S , the output R is nearly uniform even given S . Usually, we think of the source X as coming from nature and therefore consider it to be worst-case but not adversarial – this is used to justify its independence from S . DVW considered a setting where the seed S is repeatedly used to extract randomness from nature and may therefore impact nature itself (e.g., consider using the timing of interrupts to derive entropy, but the interrupts may depend on processes that may themselves rely on extracted outputs). They model this by assuming that the source which produces X can depend on oracle access to the extractor $\text{Ext}(\cdot; S)$, but is independent of the seed S otherwise, and they refer to such sources as *extractor-dependent sources*. DVW showed that extractors for extractor-dependent sources cannot exist unconditionally and at least imply one-way functions. They also distinguished between two scenarios, depending on whether the source can output

some additional correlated *auxiliary information* AUX in addition to the sample X , as long as it preserves the entropy of X . The setting with auxiliary information is considered more realistic. As their main results, DVW show how to construct extractors for extractor-dependent sources in the setting without auxiliary information from sub-exponentially secure one-way functions, and in the setting with auxiliary information from a wide range of Cryptomania assumptions such as DDH, DLIN, LWE or DCR. They also gave some evidence that it would be difficult to construct such extractors from simple Minicrypt primitives, by showing that a large class of constructions—ones where seeing the outputs of the extractor on many inputs uniquely determines the seed—cannot be proven secure via a black box reduction.

Despite the above negative result, in this work we construct extractors for extractor-dependent sources, even in the setting with auxiliary information, from standard one-way functions! Our construction does not require sub-exponential security, is entirely black-box in the one-way function, and achieves the same parameters as the prior constructions from Cryptomania assumptions. We circumvent the negative result of DVW by using a construction that lies outside of the class considered there—by relying on lossiness, we ensure that many outputs don't uniquely determine the seed—yet can still be instantiated in Minicrypt. Our main technique is to adapt a construction of DVW, which relied on all-lossy-but-one functions (without a trapdoor), and adapt it to only rely on targeted all-lossy-but-one functions.

Applications: Selective Opening Security. We also apply TLFs to the problem of selective opening security [DNRS99, BHY09] for symmetric-key encryption. A selective opening attack considers a scenario where an adversary sees a large number of ciphertexts and adaptively asks to “open” some subset of them; we would like to argue that the adversary does not learn anything about the messages encrypted in the remaining ciphertexts. An opening could correspond to seeing the encryption randomness or, if all the ciphertexts are encrypted under different keys, then seeing the corresponding secret keys. Surprisingly, selective opening security does not follow generically from standard encryption security [BDWY12, HR14, HRW16]. On the other hand, we have constructions of selective-opening secure public-key encryption for both randomness-opening and key-opening under many specific public-key assumptions [BHY09, FHKW10, HLOV11, Hof12, HPW15]. However, the problem does not appear to have been studied in the symmetric-key setting. One piece of good news is that symmetric-key encryption schemes are often “public coin”, meaning that the encryption randomness is sent in the clear as part of the ciphertext. Such schemes are automatically secure against selective randomness-opening attacks, since the randomness is available to the adversary for free! Therefore, we focus on constructing a public-coin symmetric-key encryption that achieves security under *selective key-opening attacks*. We consider a setting where n secret keys k_1, \dots, k_n are chosen uniformly at random and the adversary is given a CPA oracle for each of these keys. In addition, the adversary gets n challenge ciphertexts, one under each key. The adversary gets to adaptively choose to open some

arbitrary subset of the n ciphertexts and receive the corresponding secret keys, and we want to argue that the messages encrypted in the remaining ciphertexts stay hidden. Formalizing this requires some care and we naturally adapt the simulation-based definition of selective security from the public-key setting. We show how to construct such selectively secure symmetric-key encryption from one-way functions via our constructions of T-ALBOs/PEFs.

Application: CCA Encryption from Injective Trapdoor Functions. The recent work of [HKW20] gave a black-box construction of CCA-secure public-key encryption from any injective trapdoor function. In this work, we give a completely different construction using targeted all-injective-but-one functions (T-AIBOs). As our final result, we get CCA-secure public-key encryption from any injective trapdoor function with a very high (strongly exponential) level of security and an injective pseudorandom generator. While our end-result is strictly worse than [HKW20] in terms of the assumptions, our construction is conceptually simple and we hope it may point to further applications and/or improvements.

1.2 Our Techniques

Basic Construction. Our basic construction of targeted lossy functions (TLFs) with lossiness $\ell = 1$ is extremely simple. Let $G : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda+1}$ be an injective pseudorandom generator (PRG), where λ is the security parameter. Let $\mathcal{H} = \{h : \{0, 1\}^{3\lambda+1} \rightarrow \{0, 1\}^{3\lambda}\}$ be a universal hash function family that compresses the input by 1 bit. We define the function family $F_{\text{fk}} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda}$ via $F_{\text{fk}}(x) = h(G(x))$, where $\text{fk} = h \in \mathcal{H}$.

The above parameters ensure that if we choose $\text{fk} = h$ randomly, then the function F_{fk} is injective with overwhelming probability. In particular, for any $x_0 \neq x_1 \in \{0, 1\}^\lambda$, the probability that $G(x_0)$ and $G(x_1)$ are a collision on h is $2^{-3\lambda}$. By taking a union bound over all such pairs x_0, x_1 , the probability of there being any collision is at most $2^{-\lambda}$.

For the lossy mode of the function, we're given some random target that we denote by x_0^* . We choose an additional random input x_1^* and “program” the hash function h so that the values $G(x_0^*)$ and $G(x_1^*)$ collide, which ensures that $F_{\text{fk}}(x_0^*) = F_{\text{fk}}(x_1^*)$. Since x_0^* and x_1^* are treated symmetrically, the tuple $(\text{fk}, y = F_{\text{fk}}(x_0^*) = F_{\text{fk}}(x_1^*))$ does not disambiguate between them, and hence preserves at least $\ell = 1$ bit of entropy in the target.

We can ensure that programming h with a collision in lossy mode is computationally indistinguishable from choosing a random h in injective mode, even given the target x_0^* . For concreteness, we consider the specific universal hash function $h_a(x) = \text{chop}(a \cdot x)$ that performs a field multiplication over $\mathbb{F}_{2^{3\lambda+1}}$ and chops off the least significant bit. Using the standard representation of field elements, this implies that for all y we have $\text{chop}(y) = \text{chop}(y + 1)$.⁵ In that case, programming h to ensure $G(x_0^*)$ collides with $G(x_1^*)$ means choosing $a = (G(x_0^*) - G(x_1^*))^{-1}$.

⁵ The addition here is over $\mathbb{F}_{2^{3\lambda+1}}$ which is of characteristic 2.

But, even if we're given the target x_0^* , the value $a = (G(x_0^*) - G(x_1^*))^{-1}$ is indistinguishable from uniform by the pseudorandomness of $G(x_1^*)$. Therefore the lossy mode of choosing a for a target x_0^* is indistinguishable from the injective mode of choosing a uniformly at random. The above summarizes the entire construction and proof of security, highlighting its simplicity!

If we don't have an injective PRG, the same construction above already achieves a relaxed form of injectivity. Namely, the injective mode of the function uniquely determines the property $P(x) = G(x)$, while the lossy mode of the function loses 1-bit of information about the same property $P(x^*) = G(x^*)$ for the target x^* .

The above only achieves lossiness of 1 bit, but can amplify the lossiness arbitrarily via parallel repetition. Given a TLF F_{fk} with 1 bit of lossiness, we define $F'_{\text{fk}'}(x_1, \dots, x_\ell) = F_{\text{fk}_1}(x_1) \parallel \dots \parallel F_{\text{fk}_\ell}(x_\ell)$ for $\text{fk}' = (\text{fk}_1, \dots, \text{fk}_\ell)$ to get ℓ bits of lossiness. While the lossiness amount can be made arbitrarily large, the lossiness rate (defined as the ratio of the lossiness ℓ to the input size) is stuck at $\frac{1}{\lambda}$ and is not improved by parallel repetition.⁶

Targeted All-Injective-But-One Functions (T-AIBOs). We can also easily extend the basic construction to get a T-AIBO. For branches $\text{tag} \in \{0, 1\}^t$, we need to define a family of functions $F_{\text{fk}, \text{tag}}(\cdot)$ such that, for a special branch tag^* , the function $F_{\text{fk}, \text{tag}^*}$ is lossy and for all other branches it is injective. We can achieve this generically from any TLF without branches where the injective function key fk is uniformly random, as is the case in our basic construction. We simply set $\text{fk} = h$ to be a pairwise-independent hash function and then apply it to the value tag to derive a function key $\hat{\text{fk}} = h(\text{tag})$ for the basic TLF; the output of $F_{\text{fk}, \text{tag}}(x)$ is then set to $F_{\hat{\text{fk}}}(x)$. We program the hash so that the special branch tag^* maps to a lossy function key $\hat{\text{fk}}^*$ for the target value x^* . The output of the hash on any other $\text{tag} \neq \text{tag}^*$ is random and independent, and therefore the resulting TLF function key $\hat{\text{fk}}$ is injective with overwhelming probability.

Targeted All-Lossy-But-One Functions (T-ALBOs). Getting T-ALBOs is more involved. Recall that we need a family of functions $F_{\text{fk}, \text{tag}}(\cdot)$ such that there is a special branch tag^* on which the function is injective and, on all other branches $\text{tag} \neq \text{tag}^*$, it is cumulatively targeted-lossy for some target x^* , meaning that the entire collection of outputs on *all* the lossy tags $(F_{\text{fk}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}$ must lose ℓ -bits of information about x^* . We start with an approach that was originally proposed by [BHK11], and later abstracted more explicitly in [DVW20], as a way of converting lossy (trapdoor) functions into all-lossy-but-one (trapdoor)

⁶ We could slightly improve the lossiness rate of the basic construction to $O(\log(\lambda))/\lambda$ by using a $t = \text{poly}(\lambda)$ -wise independent hash function and programming it to have t collisions instead of just 1 collision. This would come at the cost of a larger function key fk . This slight improvement in lossiness rate would only be of interest if we were to consider exact security. Otherwise, asymptotic polynomial/negligible security is too coarse-grained to capture this improvement since it does not even distinguish between λ and λ^ϵ for $\epsilon > 0$; in other words, in the asymptotic setting we can anyway "cheat" and make the rate as high $1/\lambda^\epsilon$ by changing the security parameter to λ^ϵ and weakening exact security accordingly.

functions in the non-targeted setting. We first describe this approach and then show how to adapt it to the targeted setting.

The basic idea of [BHK11, DVW20] is to rely on function composition. As a first step, assume we have a lossy (trapdoor) function F_{fk} where both the domain and the range are $\{0, 1\}^n$, and in particular are the same. We can use it to construct an all-lossy-but-one (trapdoor) function $\overline{F}_{\text{fk,tag}}$ with tags in $\{0, 1\}^t$. We define the function key $\text{fk} = ((\text{fk}_1^0, \text{fk}_1^1), \dots, (\text{fk}_t^0, \text{fk}_t^1))$ to consist of $2t$ function keys for the underlying lossy function and we define

$$\overline{F}_{\text{fk,tag}}(x) = (F_{\text{fk}_t^{\text{tag}_t}} \circ F_{\text{fk}_{t-1}^{\text{tag}_{t-1}}} \circ \dots \circ F_{\text{fk}_1^{\text{tag}_1}})(x)$$

where tag_i denotes the i 'th bit of tag . We set the t function keys $\text{fk}_i^{\text{tag}_i^*}$ corresponding to the injective branch tag^* to be injective and the other t function keys $\text{fk}_i^{1-\text{tag}_i^*}$ to be lossy. Since the composition of injective functions is injective, it holds that $F_{\text{fk,tag}^*}$ is an injective function. On the other hand, for any lossy branch $\text{tag} \neq \text{tag}^*$, there exists some i such that $\text{tag}_i \neq \text{tag}_i^*$ and therefore one of the functions $F_{\text{fk}_i^{\text{tag}_i}}$ applied during the computation of $F_{\text{fk,tag}}(x)$ will be lossy and lose ℓ bits of information about its input, which is the same as losing ℓ bits of information about x since its input is a permutation of x . This shows that for each lossy branch $\text{tag} \neq \text{tag}^*$, the function $F_{\text{fk,tag}}$ is individually lossy. But we can even show that the lossy branches are cumulatively lossy. This is because the only information revealed about x by all the $2^t - 1$ lossy branches cumulatively, $(F_{\text{fk,tag}}(x^*))_{\text{tag} \neq \text{tag}^*}$, can be deduced from the t values one gets by applying the first $i - 1$ injective functions followed by a lossy one in position i , for $i = 1, \dots, t$. Each such output reveals at most $n - \ell$ bits of information about x and hence in they reveal at most $t(n - \ell)$ bits in total. This gives lossiness $\ell' = n - t(n - \ell)$, which can be large if ℓ is very close to n and t is small relative to n ; e.g., if $\ell = n(1 - o(1))$ and $t = o(n)$ then $\ell' = (1 - o(1))n$. Indeed, one can get such parameters from DDH.

Unfortunately, there are several issues with applying the above approach in our case. Firstly, our basic TLF does not have the same domain and range: it maps an input in $\{0, 1\}^\lambda$ to an output in $\{0, 1\}^{3\lambda}$. This makes it difficult to even syntactically rely on the above approach. Fortunately, this is relatively easy to fix. We can redefine our basic TLF with modified parameters $F_{\text{fk}} : \{0, 1\}^{3\lambda} \rightarrow \{0, 1\}^{3\lambda}$ via $F_{\text{fk}}(x) = h(G(x))$ where now we have $x \in \{0, 1\}^{3\lambda}$, the injective PRG is of the form $G : \{0, 1\}^{3\lambda} \rightarrow \{0, 1\}^{3\lambda+1}$, and the universal hash functions are of the form $h : \{0, 1\}^{3\lambda+1} \rightarrow \{0, 1\}^{3\lambda}$. This lets us syntactically use F_{fk} in the above construction to define a family with branches. Unfortunately, now F_{fk} is no longer injective when fk is chosen in ‘‘injective mode’’. However, we can regain injectivity by first pre-processing a smaller input $x \in \{0, 1\}^\lambda$ via an injective PRG $G' : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda}$. We define the overall function with branches $\overline{F}_{\text{fk,tag}} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda}$ via:

$$\overline{F}_{\text{fk,tag}}(x) = (F_{\text{fk}_t^{\text{tag}_t}} \circ F_{\text{fk}_{t-1}^{\text{tag}_{t-1}}} \circ \dots \circ F_{\text{fk}_1^{\text{tag}_1}})(G'(x)).$$

This preserves injectivity on the branch tag^* since, with overwhelming probability, each function component $F_{\text{fk}_i^{\text{tag}^*}}$ is injective over the domain of inputs $(F_{\text{fk}_i^{\text{tag}^*}} \circ \dots \circ F_{\text{fk}_1^{\text{tag}^*}})(G'(x))$ of size 2^λ . For targeted lossiness, we can now chose each of the targeted lossy keys $\text{fk}_i^{1-\text{tag}^*}$ with the target $x^*[i] = (F_{\text{fk}_i^{\text{tag}^*}} \circ \dots \circ F_{\text{fk}_1^{\text{tag}^*}})(G'(x^*))$. This is enough to show that each lossy branch is individually targeted-lossy. Unfortunately, we don't get cumulative lossiness. Recall that the argument we employed in the previous paragraph only gave cumulative lossiness $\ell' = n - t(n - \ell)$, which was only meaningful when the initial lossiness ℓ was a large fraction of the domain size n . But in our case $\ell = 1$ and hence the above does not give us any meaningful bound on ℓ' , even for tag size $t = 2$.

To solve the above issue, we need to control the lossiness more carefully to ensure that the leakages from different lossy tags don't add up. We do so by going under the hood of our basic TLF construction. We set $x_0^* = x^*$ to be the target and choose a uniformly random and independent x_1^* . We then choose all the lossy function keys $\text{fk}_i^{1-\text{tag}^*}$ to ensure that the two values x_0^*, x_1^* collide on every lossy branch. We can do so by programming the universal hash function in the i th lossy key to ensure that it has a collision on $G(x_0^*[i]), G(x_1^*[i])$ where $x_b^*[i] = (F_{\text{fk}_i^{\text{tag}^*}} \circ \dots \circ F_{\text{fk}_1^{\text{tag}^*}})(G'(x_b^*))$. This guarantees that $F_{\text{fk}_i^{1-\text{tag}^*}}(x_0^*[i]) = F_{\text{fk}_i^{1-\text{tag}^*}}(x_1^*[i])$ and so x_0^* and x_1^* collide on every lossy branch. While this ensures cumulative lossiness, we now lose indistinguishability. The reason is that the randomness of $G(x_1^*[i])$ is used twice: once to define the lossy key $\text{fk}_i^{1-\text{tag}^*}$, and once to define $x_1^*[i + 1]$, which is used to define the lossy key $\text{fk}_{i+1}^{1-\text{tag}^*}$. Since we're reusing the same randomness, the lossy keys for different values i will appear correlated and can then be distinguished from injective keys. We can fix this issue by using two different PRGs G_0 and G_1 for the 0 functions $F_{\text{fk}_i^0}$ and the 1 functions $F_{\text{fk}_i^1}$ respectively. We need G_0, G_1 to each be injective and also to be mutually pseudorandom so that, for a random x the values $G_0(x), G_1(x)$ look like random and independent values.⁷ With this modification, we preserve indistinguishability. This is because, the lossy function key $\text{fk}_i^{1-\text{tag}^*}$ now only relies on $G^{1-\text{tag}^*}(x_1^*[i])$ while the value $x_1^*[i + 1]$ used to define $\text{fk}_{i+1}^{1-\text{tag}^*}$ only relies on $G^{\text{tag}^*}(x_1^*[i])$, and hence we can argue that the two values look random and independent.

T-ALBO with Relaxed Injectivity. We can also get a T-ALBO with relaxed injectivity by using the same construction as above with standard PRGs rather than injective PRGs, and therefore from one-way functions. The observation is that, when we program the lossy mode to ensure that the target $x^* = x_0^*$ collides with some random x_1^* on every lossy tag, it's very unlikely that x_0^* and x_1^* would collide on the injective tag, even when the PRG is not injective. Therefore, although the injective output $F_{\text{fk}, \text{tag}^*}(x^*)$ may not uniquely determine x^* , we ensure that

⁷ Equivalently, we can think of $G_0(x), G_1(x)$ as the left/right halves of a single PRG $G(x)$.

$F_{\text{fk},\text{tag}^*}(x^*)$ has 1-bit of entropy even given $(F_{\text{fk},\text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}$. In other words, the injective mode of the function reveals at least 1 bit of information about x^* that was lost by all the lossy evaluations on x^* . In fact, we can even ensure that the above holds if we shorten the output size of the function to just 1 bit (in which case, the function certainly can't be injective). We do so by applying a universal hash function with 1-bit output at the end, and programming it to ensure that $F_{\text{fk},\text{tag}^*}(x_0^*)$ and $F_{\text{fk},\text{tag}^*}(x_1^*)$ hash to different bits. This ensures 1 bit of entropy in a 1 bit output, and therefore the output of the injective branch is *uniformly random*, even given the outputs of all the lossy branches. We can amplify from 1 bit to many bits via parallel repetition.

Applications of T-ALBOs. We notice that T-ALBOs with relaxed injectivity directly give us *pseudo-entropy functions*, just by relabeling the components. We define the secret key k of the pseudo-entropy function as $k = (\text{fk}, s)$ to consist of a function key fk for a T-ALBO and a uniformly random input s for it. We then define the pseudo-entropy function $f_k(x) = F_{\text{fk},\text{tag}=x}(s)$ which interprets its input x as a branch and evaluates the T-ALBO on s . For any input x^* chosen a-priori, we can choose $k = (\text{fk}, s)$ by selecting a random s and choosing fk with the injective branch x^* and the target s . This guarantees the properties of a pseudo-entropy function: the value k chosen this way is indistinguishable from an honestly chosen k that is independent of x^* , but ensures that $f_k(x^*)$ has ℓ bits of statistical entropy even given $f_k(x)$ for all $x \neq x^*$. This shows that T-ALBOs directly give pseudo-entropy functions. In fact, this gives a pseudo-entropy function where the key k consists of a uniformly random secret component s and a public but carefully chosen component fk defined in terms of s and the point x^* on which we want to ensure statistical entropy. Conversely, a pseudo-entropy function of this form also gives a T-ALBO. By using a T-ALBO where lossiness ℓ is equal to the output size (as we showed can be done above), we can even ensure that $f_k(x^*)$ is uniformly random given $f_k(x)$ for all $x \neq x^*$.

Our applications to *leakage-resilient MACs*, *leakage-resilient symmetric-key encryption*, and *extractors for extractor-dependent sources* follow as interesting applications of pseudo-entropy functions.

For *selective-opening security*, we notice that our pseudo-entropy function has an additional feature. Not only can we ensure that $f_k(x^*)$ is uniformly random given $f_k(x)$ for all $x \neq x^*$, but for any output y we can even efficiently find a key k_y such that $f_{k_y}(x^*) = y$ and $f_{k_y}(x) = f_k(x)$ for all $x \neq x^*$. Intuitively, this additional feature gives us the ability to efficiently “equivocate”, which is used to get selective-opening security. In particular, a simulator can efficiently find a key k_y to open a challenge ciphertext to any value it wants, and k_y looks consistent even to an adversary that got access to a CPA oracle.

Application of T-AIBOs to CCA Security. We also give an application of T-AIBOs to CCA-secure encryption from any trapdoor function with a sufficient high level of security. We describe a simplified version of this result, and the main body gives a more general treatment. Let $F_{\text{fk},\text{tag}}$ be a T-AIBO with λ -bit input and $\ell = 1$ bits of lossiness, as we constructed from injective pseudorandom generators. Let f_{pk} be a family of trapdoor functions (not necessarily permutations)

with input length $n = \lambda^3$. Our CCA encryption public key consists of the pair (pk, fk) and the secret key is the trapdoor of the trapdoor function. The encryption procedure selects a random $r \in \{0, 1\}^n$ and parses it as $r = (r_1, \dots, r_d)$ with $d = \lambda^2$ and $r_i \in \{0, 1\}^\lambda$. It also selects a one-time signature key pair (vk, sk) . It computes $y = f_{pk}(r)$ and $y_1 = F_{fk, vk}(r_1), \dots, y_{\lambda^2} = F_{fk, vk}(r_d)$ then uses a Goldreich-Levin hardcore bit of r to one-time pad the message and signs everything under vk . The decryption procedure checks the signature, inverts y to recover r and checks that y_1, \dots, y_d were computed correctly: if so it recovers the hardcore bit and decrypts the message, else it rejects.

To prove CCA security, we select fk to be lossy on the branch $tag = vk$ and the target value r that correspond to the challenge ciphertext. We can then simulate the decryption procedure without knowing the trapdoor td by brute-force inverting all the values $y_i = F_{fk, vk}(r_i)$ in $\tilde{O}(2^\lambda)$ time. In the challenge ciphertext, the value $r = (r_1, \dots, r_d)$ has $d = \lambda^2$ bits of entropy even given y_1, \dots, y_d . We argue that this makes it hard to recover r even given the trapdoor function output $f_{pk}(r)$ and the ability to run in $\tilde{O}(2^\lambda)$ time. We show that this follows from very strong exponential hardness of the trapdoor function: we need to assume that for input length $n = \lambda^3$ no adversary running in time $\tilde{O}(2^\lambda)$ can invert the function with better than $\frac{2^{\lambda^2}}{2^{\lambda^3}}$ probability. While this is a strong assumption, note that the trivial attack that tries 2^λ random inputs only has success probability $\frac{2^\lambda}{2^{\lambda^3}}$ and generic non-uniform attacks [DGK17] can't do better than $\frac{2^{\tilde{O}(\lambda)}}{2^{\lambda^3}}$.

1.3 Relation to Distributed Point Functions

We observe an interesting connection between T-ALBOs (with relaxed injectivity), pseudo-entropy functions, and distributed-point functions (DPFs) [GI14, BGI15]. In fact, even though the notions look very different and were introduced with different goals in mind, they are essentially equivalent. We already discussed the connection between T-ALBOs and pseudo-entropy functions, and so we now show the connection to DPFs.

Distributed point functions were defined in the context of 2-server private information retrieval (PIR). They consist of a function family $f_k : [N] \rightarrow \{0, 1\}$. Given some target $x^* \in [N]$, it should be possible to choose two keys k_0, k_1 such that $f_{k_0}(x^*) \neq f_{k_1}(x^*)$ differ on the target point, but for all other points $x \neq x^*$ they are the same $f_{k_0}(x) = f_{k_1}(x)$. Each of the keys k_0, k_1 should individually computationally hide the value x^* . This gives 2-server PIR. When a client wants to retrieve a value $DB[x^*]$ at the location $x^* \in [N]$ of a database $DB \in \{0, 1\}^N$, it chooses the two keys k_0, k_1 using a target x^* and sends k_b to server b . Each server b computes $y_b = \bigoplus_{x \in [N]} f_{k_b}(x) \cdot DB[x]$ and the client computes $y_0 \oplus y_1 = DB[x^*]$. Neither server individually learns anything about the location x^* by the hiding property of the DPF.

A pseudo-entropy function with 1-bit output and 1-bit entropy almost already gives a DPF. If we select the key k to preserve entropy on x^* , then $f_k(x^*)$ has 1 bit of entropy even given $f_k(x)$ for all $x \neq x^*$. That means that there must be some key k' such that $f_{k'}(x^*) \neq f_k(x^*)$ but $f_{k'}(x) = f_k(x)$ for all $x \neq x^*$. We can define $k_0 = k$ and $k_1 = k'$ to get the two DPF keys for the point x^* . The only difficulty is ensuring that we can kind k' efficiently. Recall that in our construction of pseudo-entropy functions for T-ALBOs, we set $k = (\text{fk}, s)$ where fk is a function key of a T-ALBO with the “injective” branch x^* and the target input s . When we choose fk , our T-ALBO construction in turn sets $s_0 = s$, picks a random s_1 and ensures that the function outputs collide on s_0, s_1 for all branches $x \neq x^*$ but differ on the branch x^* . Therefore, we can efficiently set $k_0 = (\text{fk}, s_0)$ and $k_1 = (\text{fk}, s_1)$.

Interestingly, although our construction of T-ALBOs was initially inspired by the works of [BHK11, DVW20], we observe in retrospect that it is very similar to the construction of DPFs in [BG15]. Indeed the function composition construction of T-ALBOs using two PRGs G_0, G_1 is similar to the GGM construction of PRFs from PRGs [GGM86], and the use of hash functions h is similar to the use of “correction words” in the adaptation of GGM to DPFs in [BG15].

We hope that the connections between all these notions help foster a better understanding of each of them. The fact that completely different motivations and construction approaches surreptitiously converged to yield related notions and constructions should perhaps be viewed as a good indication of just how fundamental these ideas are.

2 Preliminaries

Basic Notation. For an integer N , we let $[N] := \{1, 2, \dots, N\}$. For a set S we let $x \leftarrow S$ denote sampling x uniformly at random from S . For a distribution \mathcal{S} we let $x \leftarrow \mathcal{S}$ denote sampling x according to the distribution. We will denote the security parameter by λ . We say a function $f(\lambda)$ is negligible, denoted $f(\lambda) = \text{negl}(\lambda)$, if $f(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$. A function is $f(\lambda)$ is polynomial, denoted $f(\lambda) = \text{poly}(\lambda)$, if $f(\lambda) = O(\lambda^c)$ for some constant $c > 0$. We say that an event occurs with overwhelming probability if it holds with probability $1 - \text{negl}(\lambda)$. For a randomized algorithm A , we will sometimes explicitly denote the randomness coins it uses, writing $A(x; \text{coins})$. We will write $D_1 \stackrel{c}{\approx} D_2$ if the (ensembles of) distributions D_1 and D_2 are computationally indistinguishable.

Information Theory. For two random variables X, Y with support $\text{supp}(X)$ and $\text{supp}(Y)$ respectively, we define their statistical distance $\text{SD}(X, Y)$ as

$$\text{SD}(X, Y) := \sum_{u \in \text{supp}(X) \cup \text{supp}(Y)} \frac{1}{2} |\Pr[X = u] - \Pr[Y = u]|.$$

Two ensembles of random variables $X = \{X_\lambda\}_\lambda, Y = \{Y_\lambda\}_\lambda$ are statistically close if $\text{SD}(X_\lambda, Y_\lambda) = \text{negl}(\lambda)$.

The min-entropy $H_\infty(X)$ of a random variable X is defined as

$$H_\infty(X) := -\log\left(\max_{x \in \text{supp}(X)} Pr[X = x]\right).$$

Following Dodis et al. [DORS08], we define the (average) conditional min-entropy of X given Y as: $H_\infty(X|Y) = -\log\left(\mathbb{E}_{y \leftarrow Y} \left[2^{-H_\infty(X|Y=y)}\right]\right)$. Note that $H_\infty(X|Y) = k$ iff the optimal strategy for guessing X given Y succeeds with probability 2^{-k} .

3 Definitions

3.1 Targeted Lossy Functions (TLFs)

We first define our basic notion of targeted lossy functions (TLF).

Definition 1 (Targeted Lossy Functions). *A targeted lossy function (TLF) function family with input length $n = n(\lambda)$, output length $m \geq n$ and lossiness parameter $\ell = \ell(\lambda)$ consists of PPT algorithms (InjectiveGen, LossyGen, F) with the following syntax:*

- $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$: generates a function key fk .
- $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, x^*)$: on input a target value $x^* \in \{0, 1\}^n$, generates a function key fk .
- $y = F_{\text{fk}}(x)$: a deterministic algorithm which, on input fk along with a value $x \in \{0, 1\}^n$, outputs $y \in \{0, 1\}^m$.

We require the following properties:

Injectivity: With overwhelming probability over the choice of $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$, the function F_{fk} is injective over its domain $\{0, 1\}^n$.

ℓ -Lossiness: For random variables $x^* \leftarrow \{0, 1\}^n$, $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, x^*)$, we have

$$H_\infty(x^* \mid \text{fk}, F_{\text{fk}}(x^*)) \geq \ell.$$

Indistinguishability: For all $x^* \in \{0, 1\}^n$, we have the computational indistinguishability

$$(x^*, \text{fk}_{inj}) \stackrel{c}{\approx} (x^*, \text{fk}_{loss})$$

where $\text{fk}_{inj} \leftarrow \text{InjectiveGen}(1^\lambda)$ and $\text{fk}_{loss} \leftarrow \text{LossyGen}(1^\lambda, x^*)$.

Relaxing Injectivity. We can also define a variant of TLFs with *relaxed injectivity*, where we require the injective mode to only uniquely determine some property $P(x)$ while lossy mode loses ℓ -bits of information on $P(x^*)$ for the target x^* . In particular, we require that there exists some function $P : \{0, 1\}^* \rightarrow \{0, 1\}^*$ for which the following holds:

- *Relaxed Injectivity:* With overwhelming probability over the choice of $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$ it holds that for all $x, x' \in \{0, 1\}^n$ if $F_{\text{fk}}(x) = F_{\text{fk}}(x')$ then $P(x) = P(x')$.
- ℓ -Lossiness: For random variables $x^* \leftarrow \{0, 1\}^n$, $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, x^*)$, we have

$$H_\infty(P(x^*) \mid \text{fk}, F_{\text{fk}}(x^*)) \geq \ell.$$

3.2 Targeted All-Lossy-But-One Functions (T-ALBO)

Next, we define a tagged version of TLFs, that we name targeted all-lossy-but-one functions (T-ALBO).

Definition 2 (T-ALBO). *A targeted all-lossy-but-one (T-ALBO) function family with input length $n = n(\lambda)$, output length $m \geq n$, tag length $t = t(\lambda)$ and lossiness parameter $\ell = \ell(\lambda)$, consists of PPT algorithms (InjectiveGen, LossyGen, F) with the following syntax:*

- $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$: generates a function key fk .
- $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$: on input $\text{tag}^* \in \{0, 1\}^t, x^* \in \{0, 1\}^n$, generates a function key fk .
- $y = F_{\text{fk}, \text{tag}}(x)$: a deterministic algorithm, which, on input fk, tag along with a value $x \in \{0, 1\}^n$, outputs $y \in \{0, 1\}^m$.

We require the following properties:

Injectivity: *With overwhelming probability over the choice of $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$, for all $\text{tag} \in \{0, 1\}^t$, the function $F_{\text{fk}, \text{tag}}$ is injective over the domain $\{0, 1\}^n$. Moreover, for any tag^*, x^* , with overwhelming probability over the choice of $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$, the function $F_{\text{fk}, \text{tag}^*}$ on tag tag^* is injective.*

ℓ -Lossiness: *For all $\text{tag}^* \in \{0, 1\}^t$ and random variables $x^* \leftarrow \{0, 1\}^n, \text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$, we have $H_\infty(x^* \mid \text{fk}, (F_{\text{fk}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}) \geq \ell$. We use $(F_{\text{fk}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}$ to denote the (ordered) list of outputs of the function $F_{\text{fk}, \text{tag}}(x^*)$ on all $2^t - 1$ possible lossy tags $\text{tag} \neq \text{tag}^*$.*

Indistinguishability: *For all $\text{tag}^* \in \{0, 1\}^t$ and all $x^* \in \{0, 1\}^n$, we have*

$$(\text{tag}^*, x^*, \text{fk}_{\text{inj}}) \stackrel{c}{\approx} (\text{tag}^*, x^*, \text{fk}_{\text{loss}})$$

where $\text{fk}_{\text{inj}} \leftarrow \text{InjectiveGen}(1^\lambda)$ and $\text{fk}_{\text{loss}} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$.

Relaxing Injectivity: Entropy-Preserving T-ALBOs. We can also relax injectivity in much the same way as we did for TLFs, by requiring that injective mode uniquely determines some property $P(x)$ while lossy mode loses information on $P(x^*)$. Here, we can even allow the property P to depend on fk, tag^* . In this case, without loss of generality, we can set $P(x) = F_{\text{fk}, \text{tag}^*}(x)$ to be the output on the “injective” tag, and therefore it tautologically holds that $F_{\text{fk}, \text{tag}^*}(x)$ determines $P(x)$. Hence this notion just requires that seeing the output of the function on input x^* over all lossy branches $\text{tag} \neq \text{tag}^*$ preserves some entropy of the output $F_{\text{fk}, \text{tag}^*}(x^*)$ on the “injective” branch tag^* . We call this notion *entropy preserving*. This notion also meaningfully allows us to make the output much smaller than the input size, and potentially just 1-bit.

Definition 3 (Entropy-Preserving T-ALBO). *An entropy-preserving T-ALBO with input length $n = n(\lambda)$, output length $m = m(\lambda)$, tag length $t = t(\lambda)$ and lossiness parameter $\ell = \ell(\lambda)$, consists of algorithms (InjectiveGen, LossyGen, F).*

We require that they satisfy the same indistinguishability property as in Definition 2. However, we replace the injectivity and lossiness properties with the following entropy-preserving property. For any fixed $\text{tag}^* \in \{0, 1\}^t$ we have:

$$H_\infty(F_{\text{fk}}(\text{tag}^*, x^*) \mid \text{fk}, (F_{\text{fk}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*}) \geq \ell,$$

where we define the random variables $x^* \leftarrow \{0, 1\}^n$, $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$.

We say that $(\text{InjectiveGen}, \text{LossyGen}, F)$ is maximally entropy-preserving if $\ell = m$, where m is the output size of the T-ALBO.

3.3 Targeted All-Injective-But-One Functions (T-AIBO)

Last, we define another tagged variant of TLFs that we call *targeted all-injective-but-one lossy functions* (T-AIBO). In a T-AIBO, the branches $\text{tag} \neq \text{tag}^*$ are injective, whereas only tag^* corresponds to a lossy branch.

Definition 4 (T-AIBO). A targeted all-injective-but-one T-AIBO function family with input length $n = n(\lambda)$, output length $m \geq n$, tag length $t = t(\lambda)$ and lossiness parameter $\ell = \ell(\lambda)$, consists of PPT algorithms $(\text{InjectiveGen}, \text{LossyGen}, F)$ with the following syntax:

- $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$: generates a function key fk .
- $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$: on input $\text{tag}^* \in \{0, 1\}^t$, $x^* \in \{0, 1\}^n$, generates a function key fk .
- $y = F_{\text{fk}, \text{tag}}(x)$: a deterministic polynomial time algorithm, which, on input fk, tag along with a value $x \in \{0, 1\}^n$ outputs $y \in \{0, 1\}^m$.

We require the following properties:

Injectivity on injective branches: With overwhelming probability over the choice of $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$, we have that for all tags $\text{tag} \in \{0, 1\}^t$, the function $F_{\text{fk}, \text{tag}}$ is injective over the domain $\{0, 1\}^n$. Moreover, for any tag^*, x^* , with overwhelming probability over $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$, we have that for all tags $\text{tag} \neq \text{tag}^*$, the function $F_{\text{fk}, \text{tag}}$ is injective.

ℓ -Lossiness: For any $\text{tag}^* \in \{0, 1\}^t$ and random variables $x^* \leftarrow \{0, 1\}^n$, $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$, we have $H_\infty(x^* \mid \text{fk}, F_{\text{fk}, \text{tag}^*}(x^*)) \geq \ell$.

Indistinguishability: For any $\text{tag}^* \in \{0, 1\}^t$ and $x^* \in \{0, 1\}^n$, we have the computational indistinguishability

$$(\text{tag}^*, x^*, \text{fk}_{\text{inj}}) \stackrel{c}{\approx} (\text{tag}^*, x^*, \text{fk}_{\text{loss}})$$

where $x^* \leftarrow \{0, 1\}^n$, $\text{fk}_{\text{inj}} \leftarrow \text{InjectiveGen}(1^\lambda)$ and $\text{fk}_{\text{loss}} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$.

We could also relax injectivity as we did for TLFs and T-ALBOs. Since we do not consider this notion in the paper, we omit it for simplicity.

4 Constructions

In this section we present our constructions of TLFs and its variants. In Sect. 4.1, we give the construction of basic TLFs (Theorem 1). Then, we show in Sect. 4.2 our construction of a T-AIBO (Theorem 2). Finally, in Sect. 4.3, we build both a T-ALBO (Theorem 3) and a maximally entropy-preserving T-ALBO (Theorem 4).

4.1 Targeted Lossy Functions

We start with our base construction of TLF. We prove the following:

Theorem 1 (TLFs from Injective PRGs). *Let $\ell = \ell(\lambda)$ be a polynomial. Assuming the existence of injective PRGs, there exists a TLF with input length $n = \ell\lambda$, output length $m = 3\ell\lambda$ and lossiness ℓ .*

Let $G : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda+1}$ be an injective PRG. Let $\mathbb{F} = \mathbb{F}_{2^{3\lambda+1}}$ be the field of size $2^{3\lambda+1}$. We represent elements of \mathbb{F} in the standard manner as $3\lambda + 1$ coefficients of polynomials in $\mathbb{F}_2[X]/(f)$ for some appropriate polynomial f , so that adding elements in \mathbb{F} can be performed by adding their coefficients component wise. For $a \in \mathbb{F}$, represented as a bit string of length $3\lambda + 1$, define $\text{chop}(a)$ as the first 3λ bits of the representation of a (i.e., the last bit, corresponding to the constant term of the polynomial, is chopped off). Let $e = 1_{\mathbb{F}} \in \mathbb{F}$ be the field element that has 0s in the first 3λ positions and 1 in the last position. Note that for all $x \in \mathbb{F}$ we have $\text{chop}(x + e) = \text{chop}(x)$, and hence for all x_1, x_2 , $\text{chop}(x_1) = \text{chop}(x_2)$ if and only if $x_1 = x_2$ or $x_1 = x_2 + e$.

We first construct an LTF (InjectiveGen, LossyGen, F) with input length $n = \lambda$ and output length $m = 3\lambda$, and lossiness 1 as follows.

- **InjectiveGen**(1^λ): Sample $a \leftarrow \mathbb{F}$ and output $\text{fk} = a$.
- **LossyGen**($1^\lambda, x^*$): On input $x^* \in \{0, 1\}^\lambda$, set $x_0^* := x^*$ and sample $x_1^* \leftarrow \{0, 1\}^\lambda \setminus \{x^*\}$. Set $a = e \cdot (G(x_0^*) - G(x_1^*))^{-1}$, and output $\text{fk} = a$.
- $F_{\text{fk}}(x) = \text{chop}(a \cdot G(x)) \in \{0, 1\}^{3\lambda}$.

Claim 1. *Suppose $G : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda+1}$ is an injective PRG. Then (InjectiveGen, LossyGen, F) is a TLF with input length $n = \lambda$, output length $m = 3\lambda$ and lossiness $\ell = 1$.*

Proof. Note that in lossy mode, a is well-defined by injectivity of G . We prove injectivity, ℓ -lossiness and indistinguishability.

Injectivity. Fix any $x_0 \neq x_1 \in \mathbb{F}$. By injectivity of G , we have $G(x_0) \neq G(x_1)$. Therefore, $F_{\text{fk}}(x_0) = F_{\text{fk}}(x_1)$ iff $\text{chop}(a \cdot G(x_0)) = \text{chop}(a \cdot G(x_1))$, which occurs iff $a = e \cdot (G(x_0) - G(x_1))^{-1}$ or $a = 0$. This happens with probability $2/|\mathbb{F}|$ over the randomness of $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$. By taking a union bound over all pairs of distinct inputs $x_0, x_1 \in \{0, 1\}^\lambda$, we obtain that the probability over $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$ that F_{fk} is not injective is at most $\frac{2^{2\lambda+1}}{|\mathbb{F}|} = \frac{1}{2^\lambda}$.

($\ell = 1$)-Lossiness. Let $x_0^* = x^* \leftarrow \{0, 1\}^\lambda$ be a random target, and let $x_1^* \leftarrow \{0, 1\}^\lambda \setminus \{x_0^*\}$ denotes the random value sampled during the execution of $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, x_0^*)$; we will denote such an execution via $\text{fk} = \text{LossyGen}(1^\lambda, x_0^*; x_1^*)$. We think of x_0^*, x_1^* as random variables, which in turn define the random variables $\text{fk} = \text{LossyGen}(1^\lambda, x_0^*; x_1^*)$, $y = F_{\text{fk}}(x_0^*) = F_{\text{fk}}(x_1^*)$. We observe that the resulting distribution of fk, y does not reveal anything about x_0^*, x_1^* beyond the (unordered) set $\{x_0^*, x_1^*\}$, due to the symmetry of how x_0^*, x_1^* are treated by LossyGen . In other words, $x_0^* \rightarrow \{x_0^*, x_1^*\} \rightarrow (\text{fk}, y)$ forms a Markov chain. A data processing inequality gives:

$$H_\infty(x_0^* \mid \text{fk}, y) \geq H_\infty(x_0^* \mid \{x_0^*, x_1^*\}) \geq 1,$$

where the last inequality follows since one cannot predict x_0^* given the (unordered) set $\{x_0^*, x_1^*\}$ with probability better than $1/2$. Note that x_0^*, x_1^* are uniformly random over $\{0, 1\}^\lambda$ conditioned on $x_0^* \neq x_1^*$.

Indistinguishability. We define a hybrid experiment where LossyGen is modified as follows:

- $\widetilde{\text{LossyGen}}(1^\lambda, x^*)$: Set $x_0^* := x^*$ and select $u_1^* \leftarrow \mathbb{F}$. If $u_1^* = G(x_0^*)$, output $\text{fk} = 0$. Otherwise output $a = e \cdot (G(x_0^*) - u_1^*)^{-1}$.

The output of $\widetilde{\text{LossyGen}}$ is indistinguishable from the output of LossyGen by PRG security of G , noting that $u_1^* = G(x_0^*)$ with negligible probability $1/2^{3\lambda+1}$ over the randomness of u_1^* alone.

Moreover, even given x^* , the output of $\widetilde{\text{LossyGen}}$ is uniformly random in \mathbb{F} over the choice of u_1^* alone, and is therefore identically distributed as the output of InjectiveGen .

Amplifying (absolute) lossiness. The construction above only have lossiness 1. We note here that we can amplify this lossiness, which gives Theorem 1.

The idea is that (absolute) lossiness can be amplified by partitioning a (longer) input into blocks and applying an independent TLF on each chunk. Suppose TLF $(\text{InjectiveGen}, \text{LossyGen}, F)$ is a TLF with input size n , output size m and lossiness ℓ . Let $k = k(\lambda)$ be a polynomial. The modified scheme is defined as follows:

- $\overline{\text{InjectiveGen}}(1^\lambda)$: For all $i \in [k]$, compute $\text{fk}_i \leftarrow \text{InjectiveGen}(1^\lambda)$. Output $\{\text{fk}_i\}_{i \in [k]}$.
- $\overline{\text{LossyGen}}(1^\lambda, x^*)$: On input $x^* \in \{0, 1\}^{kn}$, parse $x^* = x_1 \parallel \dots \parallel x_k \in \{0, 1\}^{kn}$ where $x_i \in \{0, 1\}^n$ for all $i \in [k]$. For all $i \in [k]$, compute $\text{fk}_i \leftarrow \text{LossyGen}(1^\lambda, x_i)$. Output $\{\text{fk}_i\}_{i \in [k]}$.
- $\overline{F}_{\text{fk}}(x)$: On input $x \in \{0, 1\}^{kn}$, parse $x = x_1 \parallel \dots \parallel x_k \in \{0, 1\}^{kn}$ where $x_i \in \{0, 1\}^n$ for all $i \in [k]$. For all $i \in [k]$, compute $y_i = F_{\text{fk}_i}(x_i)$. Output $(y_1 \parallel \dots \parallel y_k)$.

The resulting scheme is a TLF with input size kn , output size km and lossiness $k\ell$. This is at the cost of making the input longer, and therefore doesn't affect the lossiness rate. Applying the above to our construction from Claim 1, we obtain Theorem 1.

Remark: Relaxed Injectivity. If we take our construction of TLFs above but remove the requirement that the PRG is injective, we get relaxed injectivity with the property $P(x) = G(x)$. The proof is otherwise identical.

4.2 T-AIBOs

We describe our construction of T-AIBO. We prove the following:

Theorem 2 (T-AIBOs from Injective PRGs). *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be polynomials. Assuming the existence of injective PRGs, there exists a T-AIBO with input length $n = \ell\lambda$, tag length t , output length $m = \ell \cdot (3\lambda + t)$ and lossiness ℓ .*

We build on our construction of TLF from Sect. 4.1. Recall that we built our TLF as $F_{\text{fk}}(s) = \text{chop}(a \cdot G(s))$, where $a \in \mathbb{F}$ forms the key fk . In order to build a T-AIBO, we now compute $a_{\text{tag}} = h_k(\text{tag})$ where h is a pairwise independent hash function.

More formally, let $t = t(\lambda)$ be the tag length. Let $n = 3\lambda + t + 1$, and let $\mathbb{F} = \mathbb{F}_{2^n}$. We consider elements $\text{tag} \in \{0, 1\}^t$ as elements of \mathbb{F} (for instance by considering any injection induced by the coefficient embedding of \mathbb{F} as in Sect. 4.1, setting the remaining $3\lambda + 1$ entries as 0), over which we define the pairwise independent hash function

$$h_{u,v}(\text{tag}) = u \cdot \text{tag} + v \in \mathbb{F},$$

where $u, v \in \mathbb{F}$. We will use the following useful algorithm related to h :

- **Equivocate**(tag, y): on input $\text{tag} \in \mathbb{F}$ and $y \in \mathbb{F}$, sample $u \leftarrow \mathbb{F}$, compute $v = y - u \cdot \text{tag}$, and output (u, v) .

Namely, **Equivocate**(tag, y) samples a random key (u, v) conditioned on $h_{u,v}(\text{tag}) = y$. Note that for any fixed $\text{tag} \in \mathbb{F}$, we have that $h_{u,v}(\text{tag})$ is uniform over \mathbb{F} over the randomness of (u, v) . As a result, for all $\text{tag} \in \mathbb{F}$, we have:

$$\left(\text{tag}, (u, v) \leftarrow \mathbb{F} \times \mathbb{F}, y = h_{u,v}(\text{tag}) \right) \equiv \left(\text{tag}, (u, v) \leftarrow \text{Equivocate}(\text{tag}, y), y \leftarrow \mathbb{F} \right) \tag{1}$$

We now describe our first construction of a T-AIBO with lossiness $\ell = 1$. Let $t = t(\lambda)$ and $n = 3\lambda + t + 1$, $G : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda+t+1}$ be an injective PRG and h be the pairwise independent hash function from above. We define the following algorithms:

- **InjectiveGen**(1^λ): Sample $(u, v) \leftarrow \mathbb{F} \times \mathbb{F}$ and set $\text{fk} = (u, v)$.
- **LossyGen**($1^\lambda, \text{tag}^*, x^*$): Set $x_0^* = x^*$ and sample $x_1^* \leftarrow \{0, 1\}^\lambda \setminus \{x_0^*\}$. Let $a = e \cdot (G(x_0^*) - G(x_1^*))^{-1}$. Compute $(u, v) \leftarrow \text{Equivocate}(\text{tag}^*, a)$ and output $\text{fk} = (u, v)$.
- $F_{\text{fk}}(\text{tag}, x)$: Output $\text{chop}(h_{u,v}(\text{tag}) \cdot G(x))$.

Claim 2. *Suppose $G : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda+t+1}$ is an injective PRG. Then $(\text{InjectiveGen}, \text{LossyGen}, F)$ is a T-AIBO with input length λ , tag length t , output length $3\lambda + t$, and lossiness $\ell = 1$.*

Proof. We prove injectivity, 1-lossiness and indistinguishability.

Injectivity follows by the same argument as the TLF of Claim 1. In particular, an identical argument shows that, by injectivity of G , for any fixed $\text{tag} \in \{0, 1\}^t$, the probability that $F_{\text{fk}, \text{tag}}$ is not injective is at most $\frac{2^{2\lambda+1}}{|\mathbb{F}|} = \frac{1}{2^{t+\lambda}}$. An union bound over the 2^t possible tags shows that the probability that $F_{\text{fk}, \text{tag}}$ is not injective for some tag is at most $\frac{1}{2^\lambda}$.

The proof of 1-lossiness is identical to the proof of Claim 1. In particular, we define random variables $x_0^* = x^* \leftarrow \{0, 1\}^\lambda$ denoting the target, $x_1^* \leftarrow \{0, 1\}^\lambda \setminus \{x_0^*\}$ denoting the random value sampled during the execution of $\text{fk} \leftarrow \text{LossyGen}(1^\lambda, \text{tag}^*, x_0^*)$, and $y = F_{\text{fk}, \text{tag}^*}(x_0^*) = F_{\text{fk}, \text{tag}^*}(x_1^*)$. We observe that the resulting distribution of fk, y does not reveal anything about x_0^*, x_1^* beyond the (unordered) set $\{x_0^*, x_1^*\}$, due to the symmetry of how x_0^*, x_1^* are treated by LossyGen . In other words, $x_0^* \rightarrow \{x_0^*, x_1^*\} \rightarrow (\text{fk}, y)$ forms a Markov chain. A data-processing inequality then shows:

$$H_\infty(x_0^* \mid \text{fk}, y) \geq H_\infty(x_0^* \mid \{x_0^*, x_1^*\}) \geq 1.$$

For indistinguishability, we first argue that for any $\text{tag}^* \in \{0, 1\}^t$ and any $x^* \in \{0, 1\}^n$, the value a sampled during $\text{LossyGen}(1^\lambda, \text{tag}^*, x^*)$ is computationally indistinguishable from uniformly random by PRG security of G , over the randomness of x_1^* . Then, indistinguishability follows by Eq. (1).

As for TLFs and T-ALBOs one can amplify lossiness by concatenating T-AIBOs evaluations on blocks of the input, which gives Theorem 2.

Remark 1 (Generic Construction of T-AIBOs from TLFs). In the above, we build a T-AIBO starting from the particular TLF from Sect. 4.1. We note that our transformation above can be made semi-generic, by assuming that the injective function keys fk generated by the InjectiveGen procedure of the base TLF are (computationally indistinguishable from) uniformly random (as is the case in our construction). In that case, by mapping tag to function keys fk via a programmable pairwise independent hash function, we generically obtain a T-AIBO from such a TLF, in the same way as above.

4.3 T-ALBOs

We now describe our construction of T-ALBOs. We prove the following theorems:

Theorem 3 (T-ALBOs from Injective PRGs). *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be any polynomials. Assuming the existence of injective PRGs, there exists a T-ALBO with input length $n = \ell\lambda$, tag length t , output length $m = \ell \cdot (3\lambda + t)$ and lossiness ℓ .*

Theorem 4 (Entropy-Preserving T-ALBOs from OWFs). *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be any polynomials. Assuming the existence of one-way functions, there exists an entropy-preserving T-ALBO with input length $n = \ell\lambda$, tag length t , output length $m = \ell$ and lossiness ℓ . In particular, such a T-ALBO is maximally entropy preserving.*

Again, we build on our construction of TLF from Sect. 4.1. We refer to our technical overview for a high level overview of the following construction. We begin with our construction of a standard T-ALBO (satisfying injectivity) from any injective PRG.

Building blocks. Let $n = 3\lambda + t$. Let $\mathbb{F} = \mathbb{F}_{2^{3\lambda+t+1}}$. Let $G : \{0, 1\}^n \rightarrow \{0, 1\}^{2(n+1)}$ be a PRG. We will write $G(x) = (G^0(x), G^1(x))$. In particular, G^0 and G^1 are PRGs with input size n and output size $n + 1$; we will furthermore assume that each of the functions G^0 and G^1 is injective. We define $(\text{InjectiveGen}^0, \text{LossyGen}^0, F^0)$ and $(\text{InjectiveGen}^1, \text{LossyGen}^1, F^1)$ as follows:

- $\text{InjectiveGen}^b(1^\lambda)$: Sample $a \leftarrow \mathbb{F}$ and output $\text{fk} = a$.
- $\text{LossyGen}^b(1^\lambda, x^*, x_1^*)$: On input $x^*, x_1^* \in \{0, 1\}^n$, set $x_0^* = x^*$. If $G^b(x_0^*) = G^b(x_1^*)$, output $\text{fk} = 0$. Otherwise compute $a = e \cdot (G^b(x_0^*) - G^b(x_1^*))^{-1}$, and output $\text{fk} = a$.
- $F_{\text{fk}}^b(x) = \text{chop}(a \cdot G^b(x)) \in \{0, 1\}^n$.

Let $G' : \{0, 1\}^\lambda \rightarrow \{0, 1\}^n$ be an injective PRG.

Construction. We define a T-ALBO $(\overline{\text{InjectiveGen}}, \overline{\text{LossyGen}}, \overline{F})$ as follows.

- $\overline{\text{InjectiveGen}}(1^\lambda)$: For all $i \leq t$ and $b \in \{0, 1\}$, sample $\text{fk}_i^b \leftarrow \text{InjectiveGen}^b(1^\lambda)$, and output $\text{fk} = \{\text{fk}_i^b\}_{i \in [t], b \in \{0, 1\}}$.
- $\overline{\text{LossyGen}}(1^\lambda, \text{tag}^*, x^*)$: Sample $x_1^* \leftarrow \{0, 1\}^\lambda \setminus \{x^*\}$, and set $x_0^{(0)} = G'(x^*)$, and $x_1^{(0)} = G'(x_1^*)$.

For $i = 1$ to t , sample

$$\text{fk}_i^{\text{tag}_i^*} \leftarrow \text{InjectiveGen}^{\text{tag}_i^*}(1^\lambda).$$

Then set

$$\begin{aligned} x_0^{(i)} &= F_{\text{fk}_i^{\text{tag}_i^*}}^{\text{tag}_i^*}(x_0^{(i-1)}) = F_{\text{fk}_i^{\text{tag}_i^*}}^{\text{tag}_i^*} \circ \dots \circ F_{\text{fk}_1^{\text{tag}_1^*}}^{\text{tag}_1^*}(x_0^{(0)}) \\ x_1^{(i)} &= F_{\text{fk}_i^{\text{tag}_i^*}}^{\text{tag}_i^*}(x_1^{(i-1)}) = F_{\text{fk}_i^{\text{tag}_i^*}}^{\text{tag}_i^*} \circ \dots \circ F_{\text{fk}_1^{\text{tag}_1^*}}^{\text{tag}_1^*}(x_1^{(0)}). \end{aligned}$$

Sample

$$\text{fk}_i^{1-\text{tag}_i^*} \leftarrow \text{LossyGen}^{1-\text{tag}_i^*}(1^\lambda, x_0^{(i-1)}, x_1^{(i-1)}).$$

The output is

$$\overline{\text{fk}} = (\text{fk}_i^{(b)})_{i \in [t], b \in \{0, 1\}}.$$

– $\overline{F}_{\text{fk}, \text{tag}}(x)$: Output

$$y = F_{\text{fk}_{\text{tag}_t}}^{\text{tag}_t} \circ \dots \circ F_{\text{fk}_{\text{tag}_1}}^{\text{tag}_1}(G'(x)).$$

Claim 3. *Suppose G^0, G^1 and G' are injective PRGs, and $G = (G^0, G^1)$ is a PRG. Then $(\text{InjectiveGen}, \text{LossyGen}, \overline{F})$ is a T-ALBO with input length λ , tag length t , output length $3\lambda + t + 1$, and lossiness 1.*

Proof. We first show a useful property of $(\text{InjectiveGen}^0, \text{LossyGen}^0, F^0)$ and $(\text{InjectiveGen}^1, \text{LossyGen}^1, F^1)$.

– For all $x^* \in \{0, 1\}^n$, we have the following computational indistinguishability:

$$(\text{fk}_{\text{inj}}^0, F_{\text{fk}_{\text{inj}}^0}^0(x_1^*), x^*, \text{fk}_{\text{los}}^1) \stackrel{c}{\approx} (\text{fk}_{\text{inj}}^0, u, x^*, \text{fk}_{\text{inj}}^1), \tag{2}$$

where $\text{fk}_{\text{inj}}^0 \leftarrow \text{InjectiveGen}^0(1^\lambda)$, $\text{fk}_{\text{inj}}^1 \leftarrow \text{InjectiveGen}^1(1^\lambda)$, $x_1^* \leftarrow \{0, 1\}^n$ and $\text{fk}_{\text{los}}^1 \leftarrow \text{LossyGen}^1(1^\lambda, \text{tag}^*, x^*, x_1^*)$.

Symmetrically, we have:

$$(\text{fk}_{\text{inj}}^1, F_{\text{fk}_{\text{inj}}^1}^1(x_1^*), x^*, \text{fk}_{\text{los}}^0) \stackrel{c}{\approx} (\text{fk}_{\text{inj}}^1, u, x^*, \text{fk}_{\text{inj}}^0), \tag{3}$$

where $\text{fk}_{\text{inj}}^1 \leftarrow \text{InjectiveGen}^1(1^\lambda)$, $\text{fk}_{\text{inj}}^0 \leftarrow \text{InjectiveGen}^0(1^\lambda)$, $x_1^* \leftarrow \{0, 1\}^n$ and $\text{fk}_{\text{los}}^0 \leftarrow \text{LossyGen}^0(1^\lambda, \text{tag}^*, x^*, x_1^*)$.

These properties follow by PRG security of $G = (G^0, G^1)$, so that $F_{\text{fk}_{\text{inj}}^b}(x_1^*)$ is computationally indistinguishable from uniformly random over the randomness of $G^b(x_1^*)$, while indistinguishability of $\text{fk}_{\text{inj}}^{1-b}$ and $\text{fk}_{\text{los}}^{1-b}$ follows over the (independent) randomness of $G^{1-b}(x_1^*)$.

We now prove that the construction above is a T-ALBO.

Injectivity. Fix $x_0 \neq x_1 \in \{0, 1\}^\lambda$. By injectivity of G' , we have $G'(x_0) \neq G'(x_1)$. Let $i \in [t]$ and $b \in \{0, 1\}$, and let $x_0^{(i)} \neq x_1^{(i)} \in \{0, 1\}^n$. By injectivity of G^b , the probability over $\text{fk}_i^b \leftarrow \text{InjectiveGen}^b(1^\lambda)$ that $F_{\text{fk}_i^b}^b(x_0^{(i)}) = F_{\text{fk}_i^b}^b(x_1^{(i)})$ is $2/|\mathbb{F}|$ (which correspond to either $a = e \cdot (G^b(x_0^{(i)}) - G^b(x_1^{(i)}))^{-1}$ or $a = 0$). In particular, for any fixed $\text{tag} \in \{0, 1\}^t$, we have, by taking an union bound over $i \in [t]$, that the probability over $\text{fk} \leftarrow \text{InjectiveGen}(1^\lambda)$ that $\overline{F}_{\text{fk}, \text{tag}}(G'(x_0)) = \overline{F}_{\text{fk}, \text{tag}}(G'(x_1))$ is at most $2t/|\mathbb{F}|$. Then, by union bound over all $\text{tag} \in \{0, 1\}^t$ and pairs of input $x_0 \neq x_1 \in \{0, 1\}^\lambda$, the probability that there exists a tag tag and two inputs $x_0 \neq x_1$ such that $\overline{F}_{\text{fk}, \text{tag}}(G'(x_0)) = \overline{F}_{\text{fk}, \text{tag}}(G'(x_1))$ is at most $\frac{2t \cdot 2^t \cdot 2^{2\lambda}}{|\mathbb{F}|} = \frac{t}{2^\lambda}$, which is negligible.

An almost identical argument (without the union bound over all tags) shows that the branch tag^* is injective in lossy mode.

1-Lossiness. Fix $\text{tag}^* \in \{0, 1\}^t$. Let $x_0^* \leftarrow \{0, 1\}^\lambda$ be the target, and let $x_1^* \leftarrow \{0, 1\}^\lambda \setminus \{x_0^*\}$ be the value sampled during $\overline{\text{fk}} \leftarrow \overline{\text{LossyGen}}(1^\lambda, \text{tag}^*, x_0^*)$.

First, we claim that, for all $\text{tag} \neq \text{tag}^*$, $\overline{F}_{\overline{\text{fk}}, \text{tag}}(x_0^*) = \overline{F}_{\overline{\text{fk}}, \text{tag}}(x_1^*)$. To see this, fix any $\text{tag} \neq \text{tag}^*$, and let i denote the smallest index in $[t]$ such that $\text{tag}_i \neq \text{tag}_i^*$. Recall that we have

$$\text{fk}_i^{1-\text{tag}_i^*} \leftarrow \text{LossyGen}^{1-\text{tag}_i^*}(1^\lambda, x_0^{(i-1)}, x_1^{(i-1)}),$$

and in particular, by construction of LossyGen and F :

$$F_{\text{fk}_i^{\text{tag}_i}}(x_0^{(i-1)}) = F_{\text{fk}_i^{\text{tag}_i}}(x_1^{(i-1)}).$$

As $\text{tag}_j = \text{tag}_j^*$ for all $j < i$, we have by construction of $x_0^{(i-1)}$ and $x_1^{(i-1)}$:

$$y_i := F_{\text{fk}_i^{\text{tag}_i}} \circ \cdots \circ F_{\text{fk}_1^{\text{tag}_1}}(G'(x_0^*)) = F_{\text{fk}_i^{\text{tag}_i}} \circ \cdots \circ F_{\text{fk}_1^{\text{tag}_1}}(G'(x_1^*)),$$

and in particular

$$\begin{aligned} \overline{F}_{\overline{\text{fk}}, \text{tag}}(x_0^*) &= F_{\text{fk}_i^{\text{tag}_i}} \circ \cdots \circ F_{\text{fk}_{i+1}^{\text{tag}_{i+1}}} \circ \cdots \circ F_{\text{fk}_1^{\text{tag}_1}}(y_i) \\ &= \overline{F}_{\overline{\text{fk}}, \text{tag}}(x_1^*) \end{aligned}$$

(where by convention we consider the composition to be empty if $i = t$).

Then, we observe, similarly to the proof of Claim 1, that the resulting distribution $(\overline{\text{fk}}, (\overline{F}_{\overline{\text{fk}}, \text{tag}}(x^*))_{\text{tag} \neq \text{tag}^*})$ does not reveal anything about x_0^*, x_1^* beyond the (unordered) set $\{x_0^*, x_1^*\}$. This follows since x_0^*, x_1^* are treated symmetrically in the generation of $\overline{\text{fk}}$ and the fact that $\overline{F}_{\overline{\text{fk}}, \text{tag}}(x_0^*) = \overline{F}_{\overline{\text{fk}}, \text{tag}}(x_1^*)$ for all $\text{tag} \neq \text{tag}^*$. In other words, $x_0^* \rightarrow \{x_0^*, x_1^*\} \rightarrow (\overline{\text{fk}}, (\overline{F}_{\overline{\text{fk}}, \text{tag}}(x_0^*))_{\text{tag} \neq \text{tag}^*})$ forms a Markov chain. A data-processing inequality then shows:

$$H_\infty(x_0^* \mid \overline{\text{fk}}, (\overline{F}_{\overline{\text{fk}}, \text{tag}}(x_0^*))_{\text{tag} \neq \text{tag}^*}) \geq H_\infty(x_0^* \mid \{x_0^*, x_1^*\}) \geq 1,$$

where the last inequality follows since one cannot predict x_0^* given the (unordered) set $\{x_0^*, x_1^*\}$ with probability better than $1/2$. Note that x_0^*, x_1^* are uniformly random over $\{0, 1\}^\lambda$ conditioned on $x_0^* \neq x_1^*$.

Indistinguishability. On a high level, $x_1^{(0)}$ looks pseudorandom by security of G' . Then, we switch lossy keys to injective keys, one by one, using our special TLF property (Eq. (2), Eq. (3)), simultaneously switching $x_1^{(j)}$ to uniform and $\text{fk}_i^{1-\text{tag}_j^*}$ to injective.

Fix $\text{tag}^* \in \{0, 1\}^t$ and $x^* \in \{0, 1\}^\lambda$. We define the following hybrids:

Hybrid H_0 : This is the distribution induced by the lossy mode, namely, the output distribution is:

$$(\text{tag}^*, x^*, \overline{\text{fk}}),$$

where $\overline{\text{fk}} \leftarrow \overline{\text{LossyGen}}(1^\lambda, \text{tag}^*, x^*)$.

Hybrid $H_{1,j}$, $0 \leq j \leq t$: We switch how we generate $\overline{\text{fk}}$.

For all $i \leq t$, sample

$$\text{fk}_i^{\text{tag}_i^*} \leftarrow \text{InjectiveGen}^{\text{tag}_i^*}(1^\lambda).$$

Sample $x_1^{(j)} \leftarrow \{0, 1\}^n$, and set, for all $i > j$:

$$x_1^{(i)} = F_{\text{fk}_i^{\text{tag}_i^*}}^{\text{tag}_i^*} \circ \dots \circ F_{\text{fk}_1^{\text{tag}_{j+1}^*}}^{\text{tag}_{j+1}^*}(x_1^{(j)}),$$

and set for all $i \geq j + 1$:

$$\text{fk}_i^{1-\text{tag}_i^*} \leftarrow \text{LossyGen}^{1-\text{tag}_i^*}(1^\lambda, x_0^{(i-1)}, x_1^{(i-1)}).$$

For all $i < j$, set:

$$\text{fk}_i^{1-\text{tag}_i^*} \leftarrow \text{InjectiveGen}^{1-\text{tag}_i^*}(1^\lambda),$$

and set if $j \geq 1$:

$$\text{fk}_j^{1-\text{tag}_j^*} \leftarrow \text{InjectiveGen}^{1-\text{tag}_j^*}(1^\lambda).$$

Output $(\text{tag}^*, x^*, \overline{\text{fk}})$ where

$$\overline{\text{fk}} = (\text{fk}_i^b)_{i \in [t], b \in \{0,1\}}.$$

Note that the distribution output by Hybrid $H_{2,t}$ is identical to the one output by $\overline{\text{InjectiveGen}}(1^\lambda)$. Therefore it suffices to prove that the hybrid distributions above are indistinguishable.

Claim 4. *Assuming G' is a PRG, for all $\text{tag}^* \in \{0, 1\}^t$ and all $x^* \in \{0, 1\}^n$, the distributions*

$$(\text{tag}^*, x^*, \overline{\text{fk}})$$

generated in Hybrids H_0 and $H_{1,0}$ are computationally indistinguishable.

Proof. The only difference between these two hybrids is how $x_1^{(0)}$ is distributed. In H_0 , it is computed as $G'(x_1^*)$ where $x_1^* \leftarrow \{0, 1\}^\lambda$, and in $H_{1,0}$, as uniformly random in $\{0, 1\}^n$. Indistinguishability follows by PRG security of G' .

Claim 5. *For all $\text{tag}^* \in \{0, 1\}^t$, all $x^* \in \{0, 1\}^n$, and for all $j \in [t]$, the distributions*

$$(\text{tag}^*, x^*, \overline{\text{fk}})$$

generated in Hybrids $H_{1,j-1}$ and $H_{1,j}$ are computationally indistinguishable.

Proof. Fix $\text{tag}^* \in \{0, 1\}^t$, $x^* \in \{0, 1\}^n$, $j \in [t]$. The only differences between hybrids $H_{1,j-1}$ and $H_{1,j}$, are how $x_1^{(j)}$ and $\text{fk}_j^{1-\text{tag}_j^*}$ are generated.

To argue indistinguishability, we use our special joint indistinguishability property of $(\text{InjectiveGen}^0, \text{LossyGen}^0, F^0)$ and $(\text{InjectiveGen}^1, \text{LossyGen}^1, F^1)$ (Eq. (2), Eq. (3)).

Suppose $\text{tag}_j^* = 0$. We use the output distribution $(\text{fk}^0, u, \bar{x}^*, \text{fk}^1)$ from Eq. (2), setting $\bar{x}^* = x_0^{(j-1)} = F_{\text{fk}_{j-1}^{\text{tag}_{j-1}^*}}^{\text{tag}_{j-1}^*} \circ \dots \circ F_{\text{fk}_1^{\text{tag}_1^*}}^{\text{tag}_1^*}(x_0^{(0)})$ where $x_0^{(0)} = G'(x^*)$. We set $\text{fk}_j^0 = \text{fk}^0$, $x_1^{(j)} = u$ and $\text{fk}_j^1 = \text{fk}^1$, where we implicitly set x_1^* from Eq. (2) as $x_1^* = x_1^{(j-1)}$ (note that $x_1^{(j-1)}$ is only used to define $x_1^{(j)}$ and $\text{fk}_j^{1-\text{tag}_j^*}$ in $H_{1,j-1}$, and is not used in $H_{1,j}$). We compute all the other components as in Hybrid $H_{1,j}$.

If the distribution of Eq. (2) comes in lossy mode (meaning that fk^0 is in lossy mode), then we obtain the output distribution of Hybrid $H_{1,j-1}$. If the distribution comes in injective mode (meaning that fk^0 is in injective mode), then we obtain the output distribution of Hybrid $H_{1,j}$.

The case $\text{tag}_j^* = 1$ is almost identical, where we use Eq. (3) instead of Eq. (2).

This overall shows that for any $\text{tag}^* \in \{0, 1\}^t$ and $x^* \in \{0, 1\}^\lambda$, the distributions $(\text{tag}^*, x^*, \bar{\text{fk}})$ induced by Hybrids H_0 and $H_{1,t}$ are computationally indistinguishable, which concludes the proof.

Amplifying Lossiness. As in our construction of TLF, the construction above of T-ALBO only has lossiness 1. We note that we can also amplify lossiness for T-ALBOs, which gives Theorem 3.

We now move on to our construction of (maximally) entropy-preserving T-ALBO.

Entropy-Preserving T-ALBOs. We also construct an entropy-preserving T-ALBO, and refer to the full version for the construction and a proof.

5 Applications of T-ALBOs

We first recall in Sect. 5.1 the definition of pseudo-entropy functions (PEF) introduced by [BHK11]. We note that any entropy-preserving T-ALBO directly gives such a PEF, thus giving a construction from one-way functions (Theorem 5). Then, we describe applications of PEF, by constructing (1) extractor-dependent extractors (Theorem 6), (2) leakage-resilient, deterministic MACs (Theorem 7), and (3) leakage-resilient, public-coin symmetric encryption schemes (Theorem 8).

5.1 Pseudo-Entropy Functions

Definition 5 (Pseudo-Entropy Functions). *A pseudo-entropy function family with input length $n = n(\lambda)$, output length $m = m(\lambda)$, and lossiness parameter $\ell = \ell(\lambda)$ consists of the following PPT algorithms $(\text{Gen}, \text{LossyGen}, f)$:*

- $k \leftarrow \text{Gen}(1^\lambda)$: generates a key k .
- $k \leftarrow \text{LossyGen}(1^\lambda, x^*)$: on input $x^* \in \{0, 1\}^n$, outputs a key k .
- $y = f_k(x)$: on input $x \in \{0, 1\}^n$, deterministically outputs $y \in \{0, 1\}^m$.

We require the following properties:

ℓ -Lossiness: For all $x^* \in \{0, 1\}^n$:

$$H_\infty(f_k(x^*)) | \{f_k(x)\}_{x \neq x^*} \geq \ell$$

over the randomness of $k \leftarrow \text{LossyGen}(1^\lambda, x^*)$.

Indistinguishability: For all $x^* \in \{0, 1\}^n$:

$$\text{Gen}(1^\lambda) \stackrel{c}{\approx} \text{LossyGen}(1^\lambda, x^*).$$

Next, we show the following:

Theorem 5 (PEFs from OWFs). *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be polynomials. Assuming the existence of one-way functions, there exists a PEF with input length t , output length ℓ and lossiness ℓ .*

We refer to the full version for construction and proof of Theorem 5.

5.2 Extractor-Dependent Extractors

We show how to build ED-extractors with auxiliary information from one-way functions:

Theorem 6 (ED-Extractors from OWFs). *Assuming the existence of one-way functions there exists an ED-extractor for α -entropy sources with auxiliary information, where $\alpha = \lambda^{\Omega(1)}$.*

We refer to the full version for the definition of ED-extractors, our construction and the proof of Theorem 6.

5.3 Leakage-Resilient Symmetric-Key MACs

In the full version, we show how to build leakage-resilient symmetric-key MACs from PEFs.

Theorem 7 (Deterministic Leakage-Resilient MACs from OWFs). *Let $m = \text{poly}(\lambda)$ be a message length and $t \geq \omega(\log \lambda)$ be a tag length. Assuming one-way functions, there exists, for all L such that $t - L = \omega(\log \lambda)$, a deterministic MAC with message length m and tag length t that satisfies selective unforgeability even given leakage of size L .*

We refer to the full version for the construction and proof of Theorem 7.

5.4 Leakage-Resilient Symmetric Encryption

In the full version, we show how to build leakage-resilient symmetric encryption from PEFs.

Theorem 8 (Leakage-Resilient Symmetric Encryption from OWFs). *For any polynomial leakage amount $L = L(\lambda)$ and message length $m = m(\lambda)$, assuming one-way functions exists, there exists a public-coin, symmetric encryption scheme with message length m , ciphertext size $m + O(\lambda)$, and key size $O((L + \lambda)\lambda)$.*

We refer to the full version for the construction and proof of Theorem 8.

5.5 Symmetric-Key Encryption Secure Against Selective Opening Attacks

Finally, we show that PEFs have applications to security against selective opening attacks. We refer to the full version for a definition of selective opening security.

Theorem 9 (Selective Opening Security from OWFs). *Assuming one-way functions exist, there exists a symmetric-key encryption scheme that achieves simulation-security against selective opening of keys and randomness.*

We refer to the full version for a definition of selective opening security, and the construction and a proof of Theorem 9.

6 Application of T-AIBOs to CCA Security

We prove the following theorem:

Theorem 10 (CCA Encryption from Strong Trapdoor Functions). *Let $d = d(\lambda)$, $n = n(\lambda) = \omega(\log \lambda)$, $\rho = d \cdot n$, and $m = \max(n + 1, \lambda)$. Let TDF be a trapdoor function with input length ρ . Suppose that no time $T = 2^n \cdot \text{poly}(\lambda)$ adversary can invert TDF with probability $\frac{2^d}{2^\rho} \cdot \epsilon$ for any non-negligible ϵ . Assume furthermore the existence of an injective PRG $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$.*

Then there exists a CCA-secure (public-key) encryption scheme.

We refer to the full version for the construction and proof of Theorem 10.

References

- [ADW09] Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_3
- [AGV09] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_28
- [BBN+09] Bellare, M., et al.: Hedged public-key encryption: how to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_14
- [BDWY12] Bellare, M., Dowsley, R., Waters, B., Yilek, S.: Standard security does not imply security against selective-opening. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 645–662. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_38
- [BFO08] Boldyreva, A., Fehr, S., O’Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_19

- [BGI15] Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_12
- [BHK11] Braverman, M., Hassidim, A., Kalai, Y.T.: Leaky pseudo-entropy functions. In: Chazelle, B. (ed.) ICS 2011, pp. 353–366. Tsinghua University Press, January 2011
- [BHY09] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_1
- [CPW20] Chakraborty, S., Prabhakaran, M., Wichs, D.: Witness maps and applications. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 220–246. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_8
- [DGK17] Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: random oracles with auxiliary input, revisited. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 473–495. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_16
- [DNRS99] Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. In: 40th FOCS, pp. 523–534. IEEE Computer Society Press, October 1999
- [DORS08] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.D.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **38**(1), 97–139 (2008)
- [DVW20] Dodis, Y., Vaikuntanathan, V., Wichs, D.: Extracting randomness from extractor-dependent sources. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 313–342. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_12
- [FGK+13] Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. *J. Cryptol.* **26**(1), 39–74 (2013)
- [FHKW10] Fehr, S., Hofheinz, D., Kiltz, E., Wee, H.: Encryption schemes secure against chosen-ciphertext selective opening attacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 381–402. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_20
- [GGH19] Garg, S., Gay, R., Hajiabadi, M.: New techniques for efficient trapdoor functions and applications. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 33–63. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_2
- [GGM86] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
- [GI14] Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 640–658. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_35
- [GKK20] Garg, A., Kalai, Y.T., Khurana, D.: Low error efficient computational extractors in the CRS model. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 373–402. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_14

- [HKW20] Hohenberger, S., Koppula, V., Waters, B.: Chosen ciphertext security from injective trapdoor functions. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 836–866. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_28
- [HLOV11] Hemenway, B., Libert, B., Ostrovsky, R., Vergnaud, D.: Lossy encryption: constructions from general assumptions and efficient selective opening chosen ciphertext security. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 70–88. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_4
- [HLWW13] Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 160–176. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_10
- [Hof12] Hofheinz, D.: All-but-many lossy trapdoor functions. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 209–227. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_14
- [HPW15] Hazay, C., Patra, A., Warinschi, B.: Selective opening security for receivers. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 443–469. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_19
- [HR14] Hofheinz, D., Rupp, A.: Standard versus selective opening security: separation and equivalence results. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 591–615. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_25
- [HRW16] Hofheinz, D., Rao, V., Wichs, D.: Standard security does not imply indistinguishability under selective opening. In: Hirt, M., Smith, A. (eds.) TCC 2016, Part II. LNCS, vol. 9986, pp. 121–145. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_5
- [KOS10] Kiltz, E., O’Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_16
- [MW20] Moran, T., Wichs, D.: Incompressible encodings. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 494–523. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_17
- [NS09] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_2
- [PW08] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 187–196. ACM Press, May 2008
- [Zha16] Zhandry, M.: The magic of ELF’s. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 479–508. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_18



The t -wise Independence of Substitution-Permutation Networks

Tianren Liu¹(✉), Stefano Tessaro¹, and Vinod Vaikuntanathan²

¹ University of Washington, Seattle, WA, USA
tianrenl@uw.edu, tessaro@cs.washington.edu

² MIT, Cambridge, MA, USA
vinodv@mit.edu

Abstract. Block ciphers such as the Advanced Encryption Standard (Rijndael) are used extensively in practice, yet our understanding of their security continues to be highly incomplete. This paper promotes and continues a research program aimed at *proving* the security of block ciphers against important and well-studied classes of attacks. In particular, we initiate the study of (almost) t -wise independence of concrete block-cipher construction paradigms such as substitution-permutation networks and key-alternating ciphers. Sufficiently strong (almost) pairwise independence already suffices to resist (truncated) differential attacks and linear cryptanalysis, and hence this is a relevant and meaningful target. Our results are two-fold.

Our first result concerns substitution-permutation networks (SPNs) that model ciphers such as AES. We prove the almost pairwise-independence of an SPN instantiated with concrete S-boxes together with an appropriate linear mixing layer, given sufficiently many rounds and independent sub-keys. Our proof relies on a *characterization* of S-box computation on input differences in terms of sampling output differences from certain subspaces, and a new randomness extraction lemma (which we prove with Fourier-analytic techniques) that establishes when such sampling yields uniformity. We use our techniques in particular to prove almost pairwise-independence for sufficiently many rounds of both the AES block cipher (which uses a variant of the patched inverse function $x \mapsto x^{-1}$ as the S -box) and the MiMC block cipher (which uses the cubing function $x \mapsto x^3$ as the S -box), assuming independent sub-keys.

Secondly, we show that instantiating a key-alternating cipher (which can be thought of as a degenerate case of SPNs) with most permutations gives us (almost) t -wise independence in $t + o(t)$ rounds. In order to do this, we use the probabilistic method to develop two new lemmas, an *independence-amplification lemma* and a *distance amplification lemma*, that allow us to reason about the evolution of key-alternating ciphers.

1 Introduction

Block ciphers are among the most fundamental building blocks in cryptography, and applications demand strong pseudorandomness properties from them. However, the simplicity of widely adopted designs, such as Substitution-Permutation

Networks (SPNs), which underlie AES, is inherently at odds with the reductionist approach of provable security, as there are no clear underlying hard mathematical problems upon which security can be based. Instead, the security validation of block ciphers has gone through cryptanalysis, and considered a number of different techniques, including *linear* [41] and *differential* [5] cryptanalysis, higher-order [36] and truncated [34] differential attacks, impossible differential attacks [33], algebraic attacks [25], integral cryptanalysis [35], biclique attacks [7], and so on.

Lacking full proofs of security, the next best thing is to prove that certain relevant *classes* of attacks *cannot* possibly succeed. The more “concrete” and less “asymptotic” such a proof is, the better, and the class of attacks should be as large as possible. The most successful such effort has developed provable bounds for linear and differential cryptanalysis, starting with the seminal work of Nyberg and Knudsen [46], and culminating with fairly precise estimates for concrete block ciphers like AES (see e.g. [29–32, 48, 49]).

t -wise independence. In this paper, we move one step forward and study the (almost) t -wise independence of concrete block ciphers – namely, for a block cipher $E : \{0, 1\}^s \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, we demand that for any distinct t inputs x_1, \dots, x_t and a random key S , the distribution of

$$E(S, x_1), \dots, E(S, x_t)$$

is statistically close to that of t uniform, but distinct, n -bit strings.

This property is attractive for two reasons. First and foremost, it is potentially achievable unconditionally by a concrete design, as long as $s \geq t \cdot n$. For example, a variant of AES-128 with 11 independent round keys¹ can (potentially) be 11-wise independent. Second, t -wise independence already implies resilience against a large class of attacks that have been previously studied. Indeed, the case $t = 2$ (i.e., almost pairwise independence) already implies resilience to linear and differential cryptanalysis but also to truncated differential attacks and any other attack that exploits statistical deviations of pairs of outputs. Similarly, t -wise independence implies resilience to order $\log_2(t)$ differential attacks. One caveat with this view point is that actual cipher instances typically have fixed-length keys which do not grow with t – however, similar to prior works on analyzing simpler properties of block ciphers, and in particular expected differential probabilities, we promote the heuristic angle that properties which are true for independent keys (possibly, unconditionally) remain true (computationally) when these keys are derived via a suitable key-scheduling algorithms from a short, single key.²

We note that existing bounds on differential probabilities for ciphers such as AES *could* imply pairwise independence, if good enough, but unfortunately, the

¹ Such an “independence assumption” is common across block cipher analyses. For more, see Sect. 1.2.

² We note that the impact of key-schedules on cryptographic attacks is mostly not well understood.

current state of the art (cf. e.g. [49]) proves *upper* bounds of the order 2^{-111} for 128-bit outputs which does not imply anything about (almost) pairwise independence. Without a finer grained understanding of the difference distribution, this *could* well imply a large distance of pairs of outputs from the uniform distribution.

Scope: Substitution-Permutation Networks. Our focus in this paper is on concrete block cipher designs (which likely benefit from other security properties, such as resilience to algebraic attacks), and in particular *Substitution-Permutation Networks* (SPNs), a class for which AES is a special instance, and a generalization thereof called *Key-Alternating Ciphers* (KACs). SPNs alternate *computationally simple* rounds as follows, starting from the state being equal to the block cipher input:

1. A key-mixing step which consists of XORing the keys bit-wise with the current state;
2. A *local* non-linear step where each bit of the output depends only on a few bits of the input; Concretely, this proceeds by partitioning the n -bit state into k b -bit blocks, and applying a non-linear permutation $S : \{0, 1\}^b \rightarrow \{0, 1\}^b$ (a so-called “S-box”) to each block in parallel;
3. A linear *mixing step* is then applied to the state.

We will refer to k as the *width* and to the important special case where $b = n$ (i.e., $k = 1$) as a *Key-Alternating Cipher* (or KAC, for short). (For this case, we can omit the mixing step without loss of generality.) Most modern ciphers are SPNs (or KACs). For example, AES uses an S-box obtained from the patched inverse $x \mapsto x^{2^b-2}$ and a mixing layer alternating two simple operations (ShiftRows and MixColumns). The MiMC cipher [1] is a KAC applying the permutation $x \mapsto x^3$ to its state.

A similar viewpoint to ours was already taken by Vaudenay’s *decorrelation theory* [51], but we are unaware of any application of decorrelation to SPNs with concrete S-boxes. (In fact, this was left as an open problem.) Similarly, Hoory *et al.* [24] also suggested the use and analysis of t -wise independence, but the resulting constructions, while very elegant and simple, are far from existing practical designs, and better fit in the general *theoretical* pursuit of building t -wise independent permutations [2, 9, 28].

Our Program. This raises the following questions: If we take t -wise independence as our security goal, what are good choices for the non-linear (resp. linear) step? Which choices *provably* work and which do not? Again, we stress that our goal is to find concrete, fixed choices of these layers, without modeling the S-box as a random permutation oracle.

Our results come in two forms:

1. Results about concrete SPN instantiations of SPNs with S-boxes such as the patched inversion function, and where we prove *pairwise* independence of the resulting construction. In particular, one of our results applies to the round structure of AES, without any simplifications or idealized assumptions.

2. Existential results, which hold for most choices of P , where we prove almost t -wise independence for KACs with a number of rounds that grows with t .

Next, we provide a detailed overview of our results, and the underlying techniques. Then, we give an overview of the most relevant related work.

1.1 Our Results and Techniques

This section gives an overview of our results, and the underlying techniques.

Pairwise independence of SPNs. Our first result deals with SPNs of width k with a concrete S-box $S : \mathbb{F}_{2^b} \rightarrow \mathbb{F}_{2^b}$ (thus, $n = b \cdot k$ is the block size here). In particular we focus on the case where the S-box is $S(x) = x^{-1}$ (patched so that $0^{-1} = 0$), though the results extend to other S-boxes. Our main theorem here can be cast as follows.

Theorem (Informal). For a suitably instantiated mixing layer,³ and as long as $\frac{2k+8}{2^b} + \sqrt{k/2^b} < \frac{1}{2}$, the r -round SPN with S-box $S(x) = x^{-1}$ of width k is δ -close to pairwise independent for sufficiently large $r = r(\delta)$. In particular, if $\frac{2k+8}{2^b} + \sqrt{k/2^b} = C/2$, then $r = O(\frac{\log(1/\delta)}{\log(1/C)})$.

We briefly highlight the main ideas behind the proof and note that we will focus in particular on showing that a *three* round SPN is $O(\sqrt{k/2^b})$ -close to pairwise independent – this result will rely on a new extraction lemma, which we explain below. We then resort to an amplification result by Maurer, Pietrzak, and Renner [42] to conclude that the $(r/3)$ -fold sequential composition of the SPN is δ -close to pairwise independent, as desired.

Our analysis of the output distribution of a three-round SPN for any two distinct inputs $x \neq x'$ will take the standard (and essentially equivalent) approach of studying the distribution of the *difference* of the outputs of the two evaluations. To this end, we start with a (fixed) input difference $\Delta = x \oplus x' \neq 0^n$. Then, our first step is to show, using (mostly) algebraic properties of the field \mathbb{F}_{2^b} , that after ignoring some corner cases that happen with probability no more than $O(k/2^b)$, the input differences to the third round – denoted by V_1, \dots, V_k – satisfy jointly a very strong distributional property, namely:

any subset of them of size $k' \leq k$ has (jointly) min-entropy at least $k'(b-1)$.

For this to be true, we only need mild assumptions on the linear mixing layer. We merely require it to be described by a full-rank $k \times k$ matrix whose entries are all non-zero.

To understand the effect of the third round, at last, we resort to our extraction lemma – we want to show in particular that the distribution of the differences Z_1, \dots, Z_k , which we obtain after applying the final round of S-boxes with input

³ Our requirement is very mild, and is in particular implied by having maximum branch number, as it is in the case in many SPN analysis.

differences V_1, \dots, V_k , is very close to uniform.⁴ Imagine first that the differences Z_i are not sampled via the S-box, but rather each Z_i is sampled independently from the $(n-1)$ -dimensional sub-space orthogonal to $\{0^b, V_i\}$. (We interpret the latter as a linear subspace of \mathbb{F}_2^b , and V_i as a vector in this space.) Our extraction lemma shows that in this case, the Z_i 's are very close to uniform – the proof uses Fourier-analytic techniques.

Of course, the Z_i 's are not sampled this way – by applying the S-boxes to inputs with differences V_i – yet, the key insight is that this is almost equivalent to our sub-space representation, in that by applying a lemma of Nyberg [45] we can show that there exist permutations π, π' such that $\pi'(Z_i)$ is $O(k/2^b)$ close to a random vector sampled orthogonal to $\{0^b, \pi(V_i)\}$.

We also give a proof of a weaker bound for a two-round SPN of order $\sqrt{2^{k-b}}$. This bound could be interesting in some parameter regimes.

The AES case. Unfortunately, we cannot apply the above theorem directly to the AES round structure or the AES parameters. First off, the AES S-box combines the inverse with a \mathbb{F}_2 -affine function – it turns out this is not particularly difficult to handle (the affine function can be cast as part of the mixing). But we encounter other problems, in that the mixing layers does not satisfy the assumptions needed for the theorem to work, and the theorem does not apply when $k = 16$ and $b = 8$. Still, we can adapt our techniques to obtain a refined analysis which tells us that *six* AES rounds (with independent sub-keys) are ϵ -close to pairwise independent, for some $\epsilon < 1/2$. Then, using the MPR result in the iteration, we obtain the following result:

Theorem. $6r$ -round AES is $2^{r-1}(0.472)^r$ -close to pairwise independence.

The bound is likely far from tight, as we expect much better, but non-trivial further work seems required to obtain a substantial improvement. However, we do stress that barring the use of independent keys (which again, are common in analyses of expected differential probabilities for AES), this theorem applies to the *actual* AES structure.

Existential Results. All of the above results are about pairwise independence. It is interesting to extend them to t -wise independence for $t \geq 3$. While we leave this important question open for SPNs and concrete S-boxes, we investigate the general question whether (almost) t -wise independent constructions exist in the first place.

To this end, we employ the probabilistic method to show that there exist permutations to instantiate a $(t+1)$ -round key-alternating cipher so that it is (almost) t -wise independent. We stress that while our probabilistic argument picks such permutations at random to show their existence, these permutations can then be *fixed*.

⁴ The last mixing stage does not affect the argument – it will merely preserve uniformity by virtue of being a permutation.

Our probabilistic argument is quite involved and requires the study of martingale sequences and their concentration. Our result follows by showing two new lemmas, and employing a careful alternation between them. The first is an *independence amplification lemma* that shows how to take a KAC that is very close to t -wise independent and by adding an additional round, obtain a KAC that is somewhat close to a $t + 1$ -wise independent distribution. The second is a *distance amplification lemma* that shows how to get from a somewhat close to t -wise independent KAC to a very close to t -wise independent KAC, again by adding one round.

1.2 Perspectives and Open Problems

On Independent Keys and Other Such “Ideal” Assumptions. We remark that, to date, *all* analyses of block ciphers make ideal assumptions such as the independence of round keys and/or ideal components. For example, analyses of (iterated) Even-Mansour ciphers assume that both the construction and the adversary have *oracle access* to a random permutation P , and that P remains unqueried on an exponential number of points. This is a *highly idealized model*: a random permutation would take exponentially many bits to write down, and indeed, in the real world, P is instantiated with a concrete permutation. The proofs say nothing about what happens to the pseudorandomness of such a cipher when P is instantiated with any concrete permutation. And moreover, analyses of multi-round constructions *all* assume independent keys.

In contrast, our work continues a research program that aims to avoid such “oracle access” assumptions. This line of work, which has its roots in the work of Nyberg in the 1990s, treats the component permutations and mixing functions as concrete functions (indeed, ones that are used in block ciphers such as AES and MiMC). While proving computational pseudorandomness is way out of reach, this line of research aims to understand the security of these constructions against concrete practical attacks.

The “independent round keys” assumption is very common and rooted in the model of Markov Ciphers of Lai, Massey, and Murphy [37], and adopted by Nyberg [45] and follow-up works. The expectation is that t -wise independence becomes t -wise pseudorandomness with an appropriate instantiation of the key schedule; nevertheless, understanding the precise role of key schedules is an important open problem.

On Algebraic and Other Attacks. The research program we undertake is to study several classes of concrete, powerful, attacks against block ciphers. In particular, t -wise independence rules out an important attack vector, but the program does not stop at just t -wise independence. In particular, the two outstanding open problems that come of this work are (a) to prove t -wise independence of multi-round AES with independent round keys, for $t > 2$; and (b) to formalize and prove security against algebraic attacks. We view solving these problems as an important quest that will likely require importing analytic techniques from mathematics and TCS, as well as inventing new ones.

On Differential Attacks vs. Almost Pairwise Independence. We note that meaningful differential probabilities need to be *very* close to 2^{-n} , or else, they do not rule out distinguishers. For example, in the case of AES-128, a 2^{-127} bound on the expected differential probability (see Sect. 2 for the definition) does not rule out the first bit of the output being always the same as the first bit of the input. In this case, there is a distinguisher that always works!

We note that our analysis can make the statistical distance as small as we want with sufficiently many rounds, and in particular, make the differential probabilities arbitrarily close to the ideal 2^{-128} . We note that ours is the first such result; in particular, our result for AES is the first such optimal bound for the AES design. Showing a tighter tradeoff between the number of rounds and the statistical distance is an interesting open question. Showing a direct bound on the differential probability without going through statistical distance would be interesting as well.

1.3 Related Work

Coppersmith and Grossman [15] and Kaliski, Rivest and Sherman [26] analyzed the groups generated by transition functions of the DES block cipher. [10] show that the group generated by the round functions of a cipher similar to AES is the alternating group. On the other hand, [44] provide a cautionary tale where guarantees on the group generated by the round functions does not guarantee security.

Bounds on Linear and Differential Probabilities. There is an extensive body of literature on *provable* bounds for linear [41] and differential cryptanalysis [5] of block ciphers. We note that while sufficiently strong bounds on the differential probability – say $(1 + \epsilon)2^{-n}$ for block size n and $\epsilon = o(1)$ – would imply almost pairwise independence, these works fall short of proving such strong guarantees.

Adopting the formal framework of Lai, Massey, and Murphy [37], Nyberg and Knudsen [46] prove bounds on the differential probability for Feistel ciphers as a function of the underlying non-linear function. Several works have been devoted to studying the differential properties of fixed functions to instantiate these results – relevant to this work, [45] is the first work to show properties of differentials of the inverse permutation $x \mapsto x^{-1}$ in a finite field (these were later revisited by Daemen and Rijmen [17]). We also refer to [6] for a comprehensive survey on the progress in designing non-linear functions suitable for cryptography.

Much effort has also been devoted to provable bounds on linear and differential probabilities for AES and (more abstractly) SPNs. Hong et al. [23] gave the first analysis of two-round SPNs where the mixing layer has optimal branch number. This result was further generalized to arbitrary branch number by Kang et al. [27]. Very concrete bounds for the specific case of AES were then given via refined methods in several works [29–32, 48, 49]. The best known result here shows that the maximum expected differential probability is at most 1.144×2^{-111} for four rounds of AES. Miles and Viola [43] also provide generic bounds (i.e.,

these bounds only depend on the S-box and the number of rounds) for linear and differential attacks against multi-round SPNs – however, the quality of their bounds decreases with a higher number of rounds.

Baignères and Vaudenay [4] proved optimal resilience to differential cryptanalysis whenever the S-boxes are chosen uniformly at random and secret (i.e., their description is part of the key). Later, Miles and Viola [43] improves this result (implicitly) by showing that SPNs with random S-boxes are effectively a pseudorandom function when the number of queries is smaller than the input size of the S-box.

Stronger Differentials. Strong notions of differential attacks have been proposed. For example, Lai [36] introduced the notion of *higher order differentials*, which consider the k -th derivative (as opposed to the simple derivative of a function), whereas Knudsen [34] introduced *truncated differentials*, which only consider a subset of the bits of the output. We note that security against k -th order differential cryptanalysis is implied by the k -wise independence, whereas pairwise independence implies resistance to truncated differential cryptanalysis. Another attack technique introduced by Knudsen is that of “impossible differential attacks” [33], which leverage differences which occur with probability 0 – once again, sufficiently strong pairwise independence implicitly guarantees that differences occur with sufficiently large probability.

Decorrelation theory. Vaudenay [51] takes a similar position to ours, proving properties of block cipher constructions on a bounded number of inputs, and inferring a number of properties from these statements. The work also naturally exploits a natural connection with t -wise independence, like ours. Interestingly, Vaudenay considers a number of different distance measures for the resulting distributions, and use their properties to derive a number of results. However, we are not aware of any use of decorrelation theory about the security of SPNs or KACs with concrete permutations. Still, it would be interesting to considering distance measures from decorrelation theory in the context of our paper to improve tightness.

Analyses with Public Ideal Permutations. A substantial body of works considers analyses in models where the rounds of a KAC are (public) random permutation $P : \{0, 1\}^n \rightarrow \{0, 1\}^n$ given to the adversary. In particular, since the adversary is query-bounded, she cannot obtain the entire truth table of P and therefore, this is an idealized model. (This model is effectively capturing *generic* attacks that treat these components as a black box.) Increasingly tighter bounds for security as a pseudorandom permutation have been developed by several works [8, 12, 22, 38, 50] which assume the permutations *and* the keys are independent. Other works consider identical permutations and/or identical keys [11, 52]. The model was also considered to prove the stronger version of *indifferentiability* for key-alternating ciphers (cf. [3, 20, 21]).

The model was then adapted to SPNs by assuming that the individual S-boxes are public random permutations $\{0, 1\}^b \rightarrow \{0, 1\}^b$ [13, 14, 18, 19]. Crucially,

these results assume that the number of queries to the S -box is smaller than 2^b , which is rather unrealistic for small values of b (e.g., $b = 8$ as in AES).

2 Preliminaries

Notational Conventions. When n is a positive integer, let $[n]$ denote the set $\{1, 2, \dots, n\}$. When p is a prime or prime power, let \mathbb{F}_p denote the finite field of size p . The logarithm function \log uses base 2 by default. Probability distributions are typically denoted by calligraphic letters, e.g., \mathcal{D} . Sampling an element from \mathcal{D} is denoted by $d \leftarrow \mathcal{D}$. For any finite set S , sampling x uniformly from S is denoted by $x \leftarrow S$.

Definition 1 (Entropy). For a distribution over domain Ω whose probability mass function is p .

- Its Shannon entropy is $H(p) = -\sum_{x \in \Omega} p(x) \log(p(x))$.
- Its Min-entropy is $H_\infty(p) = -\log(\max_{x \in \Omega} p(x))$.
- Its Rényi entropy of order 2, also known as the collision entropy, is $H_2(p) = -\log(\sum_{x \in \Omega} p^2(x))$.

2.1 Almost t -wise Independent Permutations and Cryptanalysis

We review notions of almost t -wise independence, and state some connections with standard notions from the cryptanalytic literature.

Definition 2. The statistical distance (or total variation distance) between two probability distributions p and q with domain Ω is $d_{TV}(p, q) := \frac{1}{2} \cdot \sum_{x \in \Omega} |p(x) - q(x)|$. Moreover, $d_{TV}(p, q) := \sum_{x \in \Omega: p(x) > q(x)} p(x) - q(x)$.

For a two argument function $F : \{0, 1\}^m \times \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ we often write $F_K(x) = F(K, x)$, and refer to F as a *function family*. (Alternatively, we use the set notation $\mathcal{F} = \{F_K\}_{K \in \{0, 1\}^m}$ whenever more convenient.) We will be considering mostly *permutation families*, where $\ell = n$, and F_K is one-to-one for each K .

Definition 3 (close to t -wise independence). We say that a permutation family $F : \{0, 1\}^m \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is ϵ -close to t -wise independent if for all distinct $x_1, \dots, x_t \in \{0, 1\}^n$, and a uniformly random m -bit string K , the distribution of $(F_K(x_1), \dots, F_K(x_t))$ has statistical distance at most ϵ from that of t uniformly sampled distinct n -bit values (i.e., sampled without repetition).

We will use the following amplification lemma, which is due to Maurer, Pietrzak, and Renner [42].

Lemma 1 (MPR Amplification Lemma). Let F and G be ϵ - and δ -close to t -wise independent permutation families. Then, the permutation family $F \circ G$ such that $(F \circ G)_{K_1 || K_2}(x) = F_{K_1}(G_{K_2}(x))$ is $2\epsilon\delta$ -close to t -wise independent.

In particular, this implies that the permutation family F^r obtained by sequential r -fold composition of an ϵ -close to t -wise independent permutation family F is $2^{r-1}\epsilon^r$ -close to t -wise independent. We point out that for a meaningful application of this lemma, we require that $\epsilon < 1/2$.

Differential and linear cryptanalysis. For a permutation family $F : \{0, 1\}^m \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, we define the *expected differential probability* (EDP) for a given pair Δ and Δ' of non-zero input- and output-differences, as

$$\text{EDP}_F(\Delta, \Delta') = \Pr_{K, X} [F_K(X \oplus \Delta) \oplus F_K(X) = \Delta'] ,$$

where K and X are independent and uniformly distributed over the m -bit and n -bit strings, respectively. We also define $\text{MEDP}_F = \max_{\Delta, \Delta' \neq 0} \text{EDP}_F(\Delta, \Delta')$. It is easy to see that if F is ϵ -close to pairwise independent, then $\text{MEDP}_F \leq \epsilon + \frac{1}{2^n - 1}$. We note that a similar result extends to any subset of n output bits, and hence to so-called *truncated* differential probabilities.

We note that higher-order differential cryptanalysis [34, 36] generalizes differential cryptanalysis to look at higher order derivatives. It is not hard to see that almost t -wise independence will imply resistance to order- $\log_2 t$ differential cryptanalysis, as the property relies on the evaluation of the cipher on at most t inputs. We note that while (almost) t -wise independence refers to attacks that look at an arbitrary set of t inputs, an order- $\log_2 t$ differential attack looks at all inputs that lie in some $\log_2 t$ -dimensional hypercube, so a total of t inputs but they are not arbitrary.

The connection between pairwise independence and linear cryptanalysis is slightly less obvious. For more details, see the final version of our paper [40].

2.2 Key-Alternating Ciphers and Substitution Permutation Networks

A *Key Alternating Cipher* (KAC) (cf. Fig. 1) is parameterized by a block size n , number of rounds r , and a fixed permutation $P : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$. A KAC is a family of functions indexed by $r + 1$ sub-keys K_0, K_1, \dots, K_r , and defined recursively as follows:

$$\begin{aligned} F_P^{(0)}(x) &= x \oplus K_0 \\ F_{P, K_0, \dots, K_i}^{(i)}(x) &= P(F_{P, K_0, \dots, K_{i-1}}^{(i-1)}(x)) \oplus K_i . \end{aligned}$$

The family of functions is $\mathcal{F}_P := \{F_{P, K_0, \dots, K_r}^{(r)}(x) : K_i \in \mathbb{F}_2^n\}$. One can also naturally extend this to have different permutations in each round.

A *Substitution-Permutation Network* (SPN) (cf. Fig. 2) can be seen as a special case of a KAC, where $n = k \cdot b$ (we refer to k as the *width*), and the permutation P is obtained from an S-box $S : \mathbb{F}_{2^b} \rightarrow \mathbb{F}_{2^b}$ and a linear *mixing layer*, described by a matrix $M \in \mathbb{F}_{2^b}^{k \times k}$. In particular, P splits its input x into k b -bit blocks x_1, \dots, x_k , and computes first $y_i = S(x_i)$ for each i , and finally

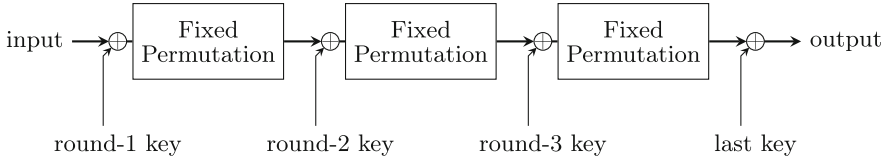


Fig. 1. Illustration of key alternating cipher

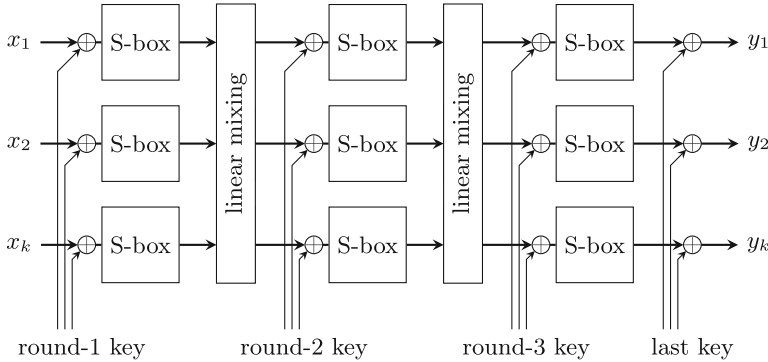


Fig. 2. Illustration of substitution permutation network

outputs $M \cdot (y_1, \dots, y_k)$. One can of course instead think of a KAC as a special of an SPN with width $k = 1$.

A fact that we will use repeatedly is that in order to bound how close to pairwise independent an SPN or KAC is, it is enough to analyze the distribution of the non-zero difference of outputs of the SPN/KAC, and its distance from the uniform distribution over non-zero strings.

Analyzing Pairwise Independence of KACs and SPNs. We will use the following lemma to reduce the analysis of pairwise independence to analyzing the distribution of differences.

Lemma 2. *Assume that the KAC (resp. SPN) \mathcal{F}_P (resp. $\mathcal{F}_{P,M}$) has the property that for any input difference $\Delta \neq 0$, the distribution of*

$$\Delta' := F_K(x) \oplus F_K(x \oplus \Delta)$$

is ϵ -close to uniform (where the randomness of the distribution is taken over x and K). Then, the KAC (resp. SPN) is ϵ -close to pairwise independent.

The proof is deferred to the full version [40].

Advanced Encryption Standard. The mostly widely used block cipher in the world is Advanced Encryption Standard (AES), which is based on the SPN framework.

The block size is 128 bits, width is 16, i.e. $n = 128, k = 16, b = 8$. AES is a family of ciphers which have 10, 12 or 14 rounds.

The S-box is instantiated by $S(x) = A(x^{2^8-2})$, where $x \mapsto x^{2^8-2}$ is the patched inverse function over \mathbb{F}_{2^8} , A is an invertible affine function over \mathbb{F}_2^8 . The exact form of A is irrelevant for this paper (as shown by Lemma 14).

The linear mixing function is instantiated by the composition of ShiftRows and MixColumns. Their descriptions are deferred to the full version [40].

2.3 Trace in Fields of Characteristic Two

We describe a number of facts related to the finite field \mathbb{F}_{2^n} of characteristic 2 and the trace function over it. For proofs of the claims below, we refer the reader to any standard text on the subject, e.g. [39].

Definition 4. *The trace function $\text{Tr} : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ is defined as $\text{Tr}(x) = \sum_{i=0}^{n-1} x^{2^i}$.*

Lemma 3. *For every $x \in \mathbb{F}_{2^n}$, $\text{Tr}(x^2) = \text{Tr}(x)$.*

Lemma 4. *For every $x, y \in \mathbb{F}_{2^n}$, $\text{Tr}(x + y) = \text{Tr}(x) + \text{Tr}(y)$. In particular, the set of elements $x \in \mathbb{F}_{2^n}$ with $\text{Tr}(x) = 0$ form an \mathbb{F}_2 -subspace of dimension $n - 1$.*

Lemma 5. *Let $\alpha \in \mathbb{F}_{2^n}$. The equation $y(y \oplus 1) = \alpha$ over \mathbb{F}_{2^n} has two solutions if $\text{Tr}(\alpha) = 0$ and no solutions otherwise.*

Corollary 1. *Let $a, b, c \in \mathbb{F}_{2^n}$ and a, b are non-zero. The equation $ax^2 + bx + c = 0$ has two solutions over \mathbb{F}_{2^n} if $\text{Tr}(ac/b^2) = 0$ and no solutions otherwise.*

Lemma 6. *For every $x \neq y \in \mathbb{F}_{2^n}$, let $S_x := \{z : \text{Tr}(xz) = 0\}$ and $S_y := \{z : \text{Tr}(yz) = 0\}$. Then, $S_x \neq S_y$. Indeed, since these are $(n - 1)$ -dimensional subspaces, they intersect at exactly 2^{n-2} elements.*

We also need the following Lemma from Nyberg’s work [45], which we reprove for completeness.

Lemma 7 ([45]). *Let $P : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ be the patched inversion function $P(x) = x^{2^n-2}$. For every $\delta, \gamma \neq 0$, let $p_{\delta, \gamma} := \Pr_{x \leftarrow \mathbb{F}_{2^n}} [P(x) \oplus P(x \oplus \delta) = \gamma]$. Then,*

$$p_{\delta, \gamma} = \begin{cases} 2/2^n, & \text{if } \delta\gamma = 1 \\ 0, & \text{if } \delta\gamma \neq 1 \end{cases} + \begin{cases} 2/2^n, & \text{if } \text{Tr}((\delta\gamma)^{-1}) = 0 \\ 0, & \text{if } \text{Tr}((\delta\gamma)^{-1}) = 1 \end{cases}$$

The following corollary is an immediate consequence.

Corollary 2. *For any non-zero $\delta \in \mathbb{F}_{2^n}$, let*

$$p(\gamma) := \Pr_{x \leftarrow \mathbb{F}_{2^n}} [P(x) \oplus P(x \oplus \delta) = \gamma] .$$

Let \mathcal{D}_δ denote the distribution with probability mass function p and let \mathcal{D}'_δ denote the distribution with probability mass function $p'(\gamma) = p(\gamma^{-1})$, we have:

- \mathcal{D}'_δ is $(2/2^b)$ -close to the uniform distribution on a subspace of dimension $b - 1$.
- $H_2(\mathcal{D}_\delta) \geq -\log_2 \left(\frac{2}{2^b} + \frac{8}{2^{2b}} \right)$.

2.4 Basics of Discrete Fourier Analysis

The characters of the group \mathbb{F}_2^n are functions $\{\chi_{\mathbf{x}} : \mathbb{F}_2^n \rightarrow \mathbb{R}\}_{\mathbf{x} \in \mathbb{F}_2^n}$ defined by

$$\chi_{\mathbf{x}}(\mathbf{y}) = (-1)^{\langle \mathbf{x}, \mathbf{y} \rangle}$$

The functions $\{\chi_{\mathbf{x}}\}_{\mathbf{x} \in \mathbb{F}_2^n}$ are orthonormal under the inner product⁵

$$\langle \chi_{\mathbf{x}}, \chi_{\mathbf{x}'} \rangle := \frac{1}{2^n} \sum_{\mathbf{y} \in \mathbb{F}_2^n} \chi_{\mathbf{x}}(\mathbf{y}) \chi_{\mathbf{x}'}(\mathbf{y}) .$$

Let $f : \mathbb{F}_2^n \rightarrow \mathbb{R}$ be a real-valued function on \mathbb{F}_2^n . Writing $f = \sum_{\mathbf{x} \in \mathbb{F}_2^n} \widehat{f}(\mathbf{x}) \chi_{\mathbf{x}}$, we have the Fourier (inversion) formulas

$$f(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} \widehat{f}(\mathbf{x}) \chi_{\mathbf{x}}(\mathbf{y}) \quad \text{and} \quad \widehat{f}(\mathbf{x}) = \langle f, \chi_{\mathbf{x}} \rangle = \frac{1}{2^n} \sum_{\mathbf{y} \in \mathbb{F}_2^n} f(\mathbf{y}) \chi_{\mathbf{x}}(\mathbf{y})$$

We need the following two facts. For proofs, we refer the reader to [47].

Lemma 8 (Parseval’s Theorem). $\frac{1}{2^n} \sum_{\mathbf{y} \in \mathbb{F}_2^n} f(\mathbf{y})^2 = \sum_{\mathbf{x} \in \mathbb{F}_2^n} \widehat{f}(\mathbf{x})^2$.

If S is a subspace of \mathbb{F}_2^n , let $S^\perp = \{\mathbf{y} : \langle \mathbf{x}, \mathbf{y} \rangle = 0 \text{ for all } \mathbf{x} \in S\}$ denote its dual subspace. If S is k -dimensional, S^\perp is $(n - k)$ -dimensional.

Lemma 9. *Let $S \subseteq \mathbb{F}_2^n$ be a subspace and f_S denote the uniform probability distribution on S . That is, $f_S(\mathbf{y}) = \frac{1}{|S|}$ if $\mathbf{y} \in S$ and 0 otherwise. Then, $\widehat{f}_S(\mathbf{x}) = \frac{1}{2^n}$ if $\mathbf{x} \in S^\perp$ and 0 otherwise.*

In particular, let $S \subseteq \mathbb{F}_2^n$ be an $(n - 1)$ -dimensional subspace which can equivalently be denoted as (the dual subspace) $S = \{0, v\}^\perp$ for some $v \in \mathbb{F}_2^n$. Then,

$$\widehat{f}_S(y) = \begin{cases} \frac{1}{2^n}, & \text{if } y \in \{0, v\} \\ 0, & \text{otherwise} \end{cases}$$

Let $f : \mathbb{F}_2^n \rightarrow \mathbb{R}, g : \mathbb{F}_2^{n'} \rightarrow \mathbb{R}$ be two real-valued functions on \mathbb{F}_2^n and $\mathbb{F}_2^{n'}$ respectively. Their tensor product $f \otimes g : \mathbb{F}_2^{n+n'} \rightarrow \mathbb{R}$ is a real-valued function on $\mathbb{F}_2^{n+n'}$ such that

$$(f \otimes g)(x, y) := f(x) \cdot g(y) \text{ for all } x \in \mathbb{F}_2^n, y \in \mathbb{F}_2^{n'} .$$

Assume X, Y are two independent random variables on \mathbb{F}_2^n and $\mathbb{F}_2^{n'}$ respectively, and f, g are the probability mass functions of X, Y . Then $f \otimes g$ is the probability mass function of (X, Y) , as

$$\Pr[(X, Y) = (x, y)] = \Pr[X = x] \cdot \Pr[Y = y] = f(x) \cdot g(y) = (f \otimes g)(x, y).$$

The Fourier transform of the tensor equals the tensor of the Fourier transforms.

Lemma 10 (Fourier transform of a Tensor). *For any $f : \mathbb{F}_2^n \rightarrow \mathbb{R}, g : \mathbb{F}_2^{n'} \rightarrow \mathbb{R}$, $\widehat{f \otimes g} = \widehat{f} \otimes \widehat{g}$.*

⁵ Note that there are two inner products at play here, one over \mathbb{F}_2^n and the other over \mathbb{R}^{2^n} , and we are abusing notation by denoting them both as $\langle \cdot, \cdot \rangle$.

3 Pairwise Independence of SPNs

The main result of this section is a proof of pairwise independence of the 3-round substitution-permutation network (see Fig. 2) where the non-linear S -box is the patched inverse function over \mathbb{F}_{2^n} , used in the AES block cipher. We will show that the 3-round SPN is ϵ -close to pairwise independent for a constant $\epsilon < 1/2$, and note that an application of the MPR amplification lemma (Lemma 1) gives us $2^{-\Omega(r)}$ -closeness to pairwise independence in $3r$ rounds.

In Sect. 3.1, we start with our main technical result, an S -box extraction lemma, which says that when the input difference of a single round of SPN has sufficient Rényi entropy, the output difference is close to uniformly random. We follow this up by describing mixing functions and their properties in Sect. 3.2. In Sect. 3.3, we then use the S -box extraction lemma and properties of mixing functions to show our main result, namely the pairwise independence of 3-round SPN. The reader is encouraged to refer back to Sect. 2.4 for relevant facts about discrete Fourier analysis as and when necessary.

3.1 The S -box Extraction Lemma

Before we state the S -box extraction lemma, we describe how it will be used to show the pairwise independence of SPNs. As noted in Lemma 2, it is sufficient to show that the distribution of *output differences* on any two inputs is close to uniformly random.

Consider the scenario in the last round of a substitution-permutation network, as illustrated in Fig. 3. Before the last round, we will show that the input difference already has high (Rényi) entropy. Indeed, we will show that if there is one round of S -boxes and mixing before the last round, Δ_i has large entropy for any $i \in [k]$; and if there are two rounds of S -boxes and mixing before the last round, the joint distribution of $(\Delta_1, \dots, \Delta_k)$ has (proportionally) high entropy. The question we ask then is, is the output (difference) vector $(\Delta'_1, \dots, \Delta'_k)$ close to uniform? The extraction lemma provides an affirmative answer to this question.

Lemma 11 (The S -Box Extraction Lemma). *Let k, b be positive integers and $n = bk$. Let \mathcal{D} be a distribution over $(\mathbb{F}_2^b)^k$ and consider the following probabilistic process called $\text{Samp}_{\mathcal{D}}$.*

1. *Sample $(v_1, \dots, v_k) \leftarrow \mathcal{D}$. Let S_1, \dots, S_k be $(b - 1)$ -dimensional subspaces where each $S_i = \{0, v_i\}^\perp$ is the subspace orthogonal to v_i .*
2. *For each $i \in [k]$, sample $x_i \leftarrow S_i$ independently at random, and output (x_1, \dots, x_k) .*

For any $T \subseteq [k]$, let v_T denote the concatenation of $(v_i)_{i \in T}$, let \mathcal{D}_T denote the distribution of v_T , let $H_2[\mathcal{D}_T]$ denote its Rényi entropy. Then, the statistical distance between the joint distribution of (x_1, \dots, x_k) and the uniform distribution over \mathbb{F}_2^{bk} is at most

$$\frac{1}{2} \sqrt{\sum_{T \subseteq [k], T \neq \emptyset} 2^{-H_2[\mathcal{D}_T]}}$$

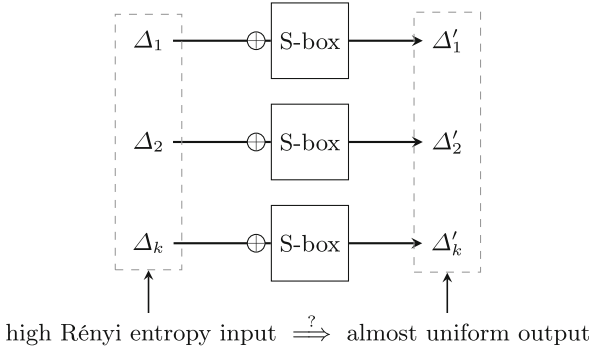


Fig. 3. Application scenario of the extraction lemma

In particular, we have:

- **Weak Extraction:** Assume that for all $i \in [k]$, $H_2[v_i] \geq h$ for a fixed real $h \leq b$. Then the statistical distance between the joint distribution of (x_1, \dots, x_k) and the uniform distribution over \mathbb{F}_2^{bk} is at most $\frac{1}{2} \cdot \sqrt{\frac{2^k - 1}{2^h}}$.
- **Strong Extraction:** Assume that for any $T \subseteq [k]$, $H_2[v_T] \geq h \cdot |T|$ where v_T denotes the concatenation of $(v_i)_{i \in T}$. Then the statistical distance between the joint distribution of (x_1, \dots, x_k) and the uniform distribution over \mathbb{F}_2^{bk} is at most

$$\frac{1}{2} \cdot \sqrt{\left(1 + \frac{1}{2^h}\right)^k - 1}$$

which, in turn, is at most $\sqrt{\frac{k}{2^{h+1}}}$ assuming $k \leq 2^h$.

Proof. Let f denote the probability mass function of $\text{Samp}_{\mathcal{D}}$. That is, $f(x_1, \dots, x_k)$ is the probability that $\text{Samp}_{\mathcal{D}}$ outputs (x_1, \dots, x_k) . Let $p(v_1, \dots, v_k)$ denote the probability assigned by the distribution \mathcal{D} to (v_1, \dots, v_k) and let ϕ_S denote the probability mass function of the uniform distribution over the subspace $S \subseteq \mathbb{F}_2^b$. Then,

$$f(x_1, \dots, x_k) = \sum_{v_1, \dots, v_k \in \mathbb{F}_2^b} p(v_1, \dots, v_k) \cdot \phi_{S_1}(x_1) \cdot \phi_{S_2}(x_2) \cdot \dots \cdot \phi_{S_k}(x_k)$$

where $S_i = \{0, v_i\}^\perp$ is an implicit function of v_i , as before. We will write this as

$$f = \sum_{v_1, \dots, v_k \in \mathbb{F}_2^b} p(v_1, \dots, v_k) \cdot \left(\phi_{S_1} \otimes \phi_{S_2} \otimes \dots \otimes \phi_{S_k}\right)$$

We are interested in the statistical distance $d_{\text{TV}}(f, u) = \frac{1}{2} \|f - u\|_1$, where u is the uniform distribution over \mathbb{F}_2^{bk} . It suffices to bound $\|\hat{f} - \hat{u}\|_2^2$ since

$$\|f - u\|_1^2 \leq 2^{kb} \|f - u\|_2^2 = 2^{2kb} \|\hat{f} - \hat{u}\|_2^2. \tag{1}$$

where the inequality comes from Cauchy-Schwartz and the equality comes from Parseval's theorem (Lemma 8).

The Fourier transform of f equals

$$\hat{f}(y_1, \dots, y_k) = \sum_{v_1, \dots, v_k \in \mathbb{F}_2^b} p_{(v_1, \dots, v_k)} \cdot \prod_{i \in [k]} \hat{\phi}_{S_i}(y_i)$$

Observe that by Lemma 9, $\hat{\phi}_{S_i}$ is 0 everywhere except for $\hat{\phi}_{S_i}(v_i) = \hat{\phi}_{S_i}(0) = 1/2^b$. Thus the only inputs (y_1, \dots, y_k) on which $\hat{f}(y_1, \dots, y_k) \neq 0$ are those in the set $\{0, v_1\} \times \{0, v_2\} \times \dots \times \{0, v_k\}$. Thus,

$$\hat{f}(y_1, \dots, y_k) = \frac{1}{2^{bk}} \cdot \Pr[v_i = y_i \text{ for all } i \text{ s.t. } y_i \neq 0]. \tag{2}$$

The ℓ_2 -norm of the Fourier transform of $f - u$ can then be computed as

$$\begin{aligned} \|\hat{f} - \hat{u}\|_2^2 &= \sum_{\substack{y_1, \dots, y_k \in \mathbb{F}_2^b \\ (y_1, \dots, y_k) \neq \mathbf{0}}} \hat{f}^2(y_1, \dots, y_k) \\ &= \sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} \sum_{\substack{y_1, \dots, y_k \in \mathbb{F}_2^b \\ y_i \neq 0 \text{ iff } i \in T}} \hat{f}^2(y_1, \dots, y_k) \\ &= \sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} \sum_{\substack{y_1, \dots, y_k \in \mathbb{F}_2^b \\ y_i \neq 0 \text{ iff } i \in T}} \frac{1}{2^{2bk}} \cdot \Pr[v_i = y_i \text{ for all } i \in T]^2. \end{aligned} \tag{3}$$

Let $v_T := (v_i)_{i \in T}$ denote the vector v restricted to indices in T , let \mathcal{D}_T denote the distribution of v_T , and let f_T denote the probability mass function of \mathcal{D}_T . Then,⁶

$$\|\hat{f} - \hat{u}\|_2^2 \leq \frac{1}{2^{2bk}} \sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} \|f_T\|_2^2 = \frac{1}{2^{2kb}} \sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} 2^{-\text{H}_2[\mathcal{D}_T]}. \tag{4}$$

Combining with equation (1) concludes the proof of the general case.

$$d_{\text{TV}}(f, u) \leq \frac{1}{2} \cdot 2^{kb} \cdot \|\hat{f} - \hat{u}\|_2 \leq \frac{1}{2} \sqrt{\sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} 2^{-\text{H}_2[\mathcal{D}_T]}}. \tag{5}$$

Setting 1: Weak Extraction. Assume for any $i \in [k]$, $\text{H}_2[\mathcal{D}_{\{i\}}] \geq h$. Then, for any non-empty set $T \subseteq [k]$, we have $\text{H}_2[\mathcal{D}_T] \geq h$. Therefore, combining with equation (5),

$$d_{\text{TV}}(f, u) \leq \frac{1}{2} \sqrt{\sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} 2^{-\text{H}_2[\mathcal{D}_T]}} \leq \frac{1}{2} \cdot \sqrt{\frac{2^k - 1}{2^h}}.$$

⁶ The first inequality symbol in the equation is tight, if V_1, \dots, V_k are always non-zero.

Setting 2: Strong Extraction. Assume for any $T \subseteq [k]$, $H_2[\mathcal{D}_T] \geq h \cdot |T|$. Then

$$\sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} 2^{-H_2[\mathcal{D}_T]} \leq \sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} \left(\frac{1}{2^h}\right)^{|T|} = \left(1 + \frac{1}{2^h}\right)^k - 1$$

using the binomial expansion. Combining with equation (5), we have

$$d_{\text{TV}}(f, u) \leq \frac{1}{2} \sqrt{\sum_{\substack{T \subseteq [k] \\ T \neq \emptyset}} 2^{-H_2[\mathcal{D}_T]}} \leq \frac{1}{2} \cdot \sqrt{\left(1 + \frac{1}{2^h}\right)^k - 1}.$$

If we additionally assume that $k \leq 2^h$, then

$$d_{\text{TV}}(f, u) \leq \frac{1}{2} \cdot \sqrt{\left(1 + \frac{1}{2^h}\right)^k - 1} \leq \frac{1}{2} \sqrt{e^{\frac{k}{2^h}} - 1} \leq \frac{1}{2} \sqrt{\frac{2k}{2^h}}.$$

The last inequality symbol holds only if $\frac{k}{2^h} \leq 1.256\dots$, which follows from the condition $k \leq 2^h$. □

We remark that Fourier analysis can be bypassed here. The above proof uses Fourier analysis to bound the collision probability. There is an alternative proof of the extraction lemma in the full version [40] that bounds the collision probability using “elementary” non-Fourier methods.

Comparing Fig. 3 with the statement of the extraction lemma. The outstanding contrast is that the extraction lemma assumes a very specific linear algebra structure. That is, consider the domain as vector space \mathbb{F}_2^b , the output (difference) vector is sampled as a random vector orthogonal to the input (difference) vector. While in each round of SPN, the input is subtracted by the random key and then feed into the S-box. The output difference *is not* sampled uniformly from a subspace.

However, we hope the two can be bridged by change of variables. Say we start with two inputs differing Δ , let Δ' denote the difference after key-subtraction and S-box. We hope there exist 1-to-1 mappings $\pi_{\text{in}}, \pi_{\text{out}} : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$ such that $\pi_{\text{out}}(\Delta')$ is a random vector orthogonal to $\pi_{\text{in}}(\Delta)$.

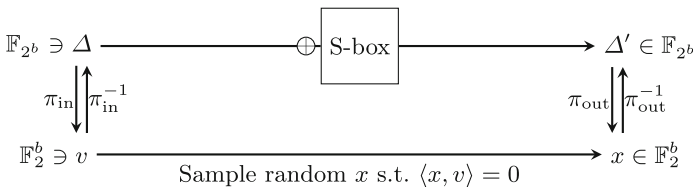


Fig. 4. Subtracting key followed by S-box \approx subspace sampling, modulo change of variables

Figure 4 illustrates the property we are looking for. Although it cannot be exactly satisfied by any S-box—we know $\pi_{\text{out}}(\Delta')$ doesn't equal x by distribution, because $\Delta = 0 \iff \Delta' = 0$ —we show that pragmatic S-boxes almost satisfy the property.

Assuming the S-box is the patched inverse function, the following lemma shows that $\pi_{\text{out}}(\Delta')$ is statistically close to a random vector orthogonal to $\pi_{\text{in}}(\Delta)$, as long as $\Delta \neq 0$.

Lemma 12. *Assume S-box is the patched inverse $P(x) = x^{2^b-2}$. There exist 1-to-1 mappings $\pi_{\text{in}}, \pi_{\text{out}} : \mathbb{F}_{2^b} \rightarrow \mathbb{F}_2^b$ such that for any non-zero $\Delta \in \mathbb{F}_{2^b}$, letting Δ' denotes a random variable defined by*

$$\Delta' := P(r) - P(r + \Delta)$$

for a uniformly random $r \in \mathbb{F}_{2^b}$, the statistical distance between $\pi_{\text{out}}(\Delta')$ and the uniform distribution over $\{0, \pi_{\text{in}}(\Delta)\}^\perp$ is no more than $\frac{2}{2^b}$.

Proof. As shown in Lemma 7 (from [45]),

$$\Pr[\Delta' = \delta] = \begin{cases} \frac{2}{2^b}, & \text{if } \delta = \frac{1}{\Delta} \\ 0, & \text{o.w.} \end{cases} + \begin{cases} \frac{2}{2^b}, & \text{if } \text{Tr}(\frac{1}{\delta\Delta}) = 0 \\ 0, & \text{o.w.} \end{cases}$$

Define $\pi_{\text{out}}(x) = x^{2^b-2}$ to be the patched inverse as well. Then

$$\Pr[\pi_{\text{out}}(\Delta') = x] = \begin{cases} \frac{2}{2^b}, & \text{if } x = \Delta \\ 0, & \text{o.w.} \end{cases} + \begin{cases} \frac{2}{2^b}, & \text{if } x \neq 0 \text{ and } \text{Tr}(\frac{x}{\Delta}) = 0 \\ 0, & \text{o.w.} \end{cases}$$

As show in Lemma 4, $x \mapsto \text{Tr}(\frac{x}{\Delta})$ is linear function over \mathbb{F}_2 . Define $\pi_{\text{in}}(\Delta)$ as the coefficient vector of $x \mapsto \text{Tr}(\frac{x}{\Delta})$. Then

$$\Pr[\pi_{\text{out}}(\Delta') = x] = \begin{cases} \frac{2}{2^b}, & \text{if } x = \Delta \\ 0, & \text{o.w.} \end{cases} + \begin{cases} \frac{2}{2^b}, & \text{if } x \neq 0 \text{ and } \langle \pi_{\text{in}}(\Delta), x \rangle = 0 \\ 0, & \text{o.w.} \end{cases}$$

Apparently, the statistical distance between $\pi_{\text{out}}(\Delta')$ and the uniform distance over $\{0, \pi_{\text{in}}(\Delta)\}^\perp$ is $\frac{2}{2^b}$. □

The following lemma shows the analogous statement for the cube function. The proof is deferred to the full version [40].

Lemma 13. *Assume S-box is the cube function $P(x) = x^3$ over \mathbb{F}_{2^b} where b is odd⁷. There exist 1-to-1 mappings $\pi_{\text{in}}, \pi_{\text{out}} : \mathbb{F}_{2^b} \rightarrow \mathbb{F}_2^b$ such that for any non-zero $\Delta \in \mathbb{F}_{2^b}$, letting Δ' denote a random variable defined by*

$$\Delta' := P(r) - P(r + \Delta)$$

for a uniformly random $r \in \mathbb{F}_{2^b}$, $\pi_{\text{out}}(\Delta')$ is the uniform distribution over $\{0, \pi_{\text{in}}(\Delta)\}^\perp$.

⁷ The condition on b being odd is necessary to ensure that P is a permutation.

In Sect. 3.4 we are going to analyze AES. The S-box in AES is called Rijndael S-box, which is not exactly the patched inverse function. Rijndael S-box is the composition of the patched inverse function and an affine transformation. The following lemma shows that the additional affine transformation makes little difference.

Lemma 14. *Assume S-box is $P(x) = A(x^{2^b-2})$, where A is an affine permutation over \mathbb{F}_2^b . There exist 1-to-1 mappings $\pi_{\text{in}}, \pi_{\text{out}} : \mathbb{F}_{2^b} \rightarrow \mathbb{F}_2^b$. For any non-zero $\Delta \in \mathbb{F}_{2^b}$, let Δ' denote a random variable defined by*

$$\Delta' := P(r) - P(r + \Delta)$$

for a uniformly random $r \in \mathbb{F}_{2^b}$. The statistical distance between $\pi_{\text{out}}(\Delta')$ and the uniform distribution over $\{0, \pi_{\text{in}}(\Delta)\}^\perp$ is no more than $\frac{2}{2^b}$.

Proof. As we are analyzing the differences, any additive constant in the affine function A has no effect. Thus we can safely assume A is a linear permutation.

When input difference is Δ , the output difference is

$$\Delta' = P(r) - P(r + \Delta) = A(r^{2^b-2}) - A((r + \Delta)^{2^b-2}) = A(r^{2^b-2} - (r + \Delta)^{2^b-2}).$$

Define $\Delta^* = r^{2^b-2} - (r + \Delta)^{2^b-2}$, then $\Delta' = A(\Delta^*)$.

Lemma 12 shows that there exists $\pi_{\text{in}}, \pi_{\text{out}}$ such that $\pi_{\text{out}}(\Delta^*)$ is close to uniform distribution over $\{0, \pi_{\text{in}}(\Delta)\}$. Define $\pi'_{\text{out}}(x) := \pi_{\text{out}}(A^{-1}(x))$. Then $\pi'_{\text{out}}(\Delta') = \pi_{\text{out}}(A^{-1}(\Delta')) = \pi_{\text{out}}(\Delta^*)$, which is close to uniform distribution over $\{0, \pi_{\text{in}}(\Delta)\}$. Thus $\pi_{\text{in}}, \pi'_{\text{out}}$ are what we need. \square

3.2 Properties of Mixing Functions

Before proceeding to show the almost-pairwise independence of SPN constructions using the extraction lemma, we describe properties that we need the mixing functions to satisfy. We define two such properties below and prove some elementary statements about them.

The first property that we call *diffusion* requires that if one of the input blocks of the (typically linear) function $M : (\mathbb{F}_{2^b})^k \rightarrow (\mathbb{F}_{2^b})^k$ has sufficient entropy and the distribution of the k input blocks are independent, then *each output block* has large entropy. It is not hard to see that both the sufficient entropy condition and the independence condition on the input are necessary for such a statement to be true. Looking ahead, this property will turn out to be useful in the first layer (or the first few layers) of the SPN where we wish to propagate differences in one input block to differences in all of them.

Property 1 (Diffusion). Let $M : (\mathbb{F}_{2^b})^k \rightarrow (\mathbb{F}_{2^b})^k$ be a function. Let $H_\alpha \in \{H_2, H_\infty\}$ be an entropy function. Let X_1, \dots, X_k be independent random variables over \mathbb{F} such that there exists an i for which $H_\alpha(X_i) \geq h$ for a real h , and let $(Y_1, \dots, Y_k) := M(X_1, \dots, X_k)$. M is *diffusing* if

$$\text{for all } i \in [k], H_\alpha(Y_i) \geq h.$$

We now show a sufficient condition for a function to be diffusing. The proof is deferred to the full version [40].

Lemma 15. *If $M \in (\mathbb{F}_{2^b})^{k \times k}$ is a matrix with no zero entry, the linear mapping $x \mapsto Mx$ is diffusing (i.e. satisfies Property 1).*

The second property that we call *entropy-preservation* requires that if all of the input blocks of the (typically linear) function $M : (\mathbb{F}_{2^b})^k \rightarrow (\mathbb{F}_{2^b})^k$ have sufficient entropy and the distribution of the k blocks are independent, then *each collection of output blocks* have large joint entropy. Looking ahead, this property will turn out to be useful in the subsequent layers of the SPN to ensure that the mixing layers do not *reduce* the entropy. As one might expect, this property comes for free if M is an invertible linear map. The proof is deferred to the full version [40].

Property 2 (Entropy Preservation). A function $M : (\mathbb{F}_{2^b})^k \rightarrow (\mathbb{F}_{2^b})^k$ is *entropy preserving* if for any entropy function $H_\alpha \in \{H_2, H_\infty\}$, for any real h , for any independent random variables X_1, \dots, X_k over \mathbb{F}_{2^b} such that $H_\alpha(X_i) \geq h$ for all $i \in [k]$, letting $(Y_1, \dots, Y_k) := M(X_1, \dots, X_k)$, we have

$$H_\alpha(Y_{i_1}, \dots, Y_{i_s}) \geq s \cdot h$$

for any $\{i_1, \dots, i_s\} \subseteq [k]$.

Lemma 16. *If $M \in (\mathbb{F}_{2^b})^{k \times k}$ is an invertible matrix, the mapping $x \mapsto Mx$ is entropy-preserving (i.e. satisfies Property 2).*

Connection to Branch Number. The branch number of a matrix $M \in (\mathbb{F}_{2^b})^{k \times k}$ is defined to be

$$\text{br}(M) = \max_{\alpha \in (\mathbb{F}_{2^b})^k} (\text{wt}(\alpha) + \text{wt}(M\alpha))$$

where wt denotes the Hamming weight. Having an optimal branch number is considered a desirable feature for mixing functions [16, 27]. An observation by Miles and Viola [43] says that any matrix with the maximal branching number of $k + 1$ also satisfies Properties 1 and 2, although the converse does not necessarily hold.

3.3 Proofs of Pairwise Independence

In this section, we show several proofs of pairwise independence of SPNs using the patched inverse function $P(x) = x^{2^b-2}$ over the finite field \mathbb{F}_{2^b} . The first result (Theorem 1) applies in a regime where $k \leq b$ is relatively small; here, the result says that a 2-round SPN is close to pairwise independent. The second result (Theorem 2) is much more general and applies to large k as long as $k \leq 2^{b-4}$; here, the result says that a 3-round SPN is close to pairwise independent.

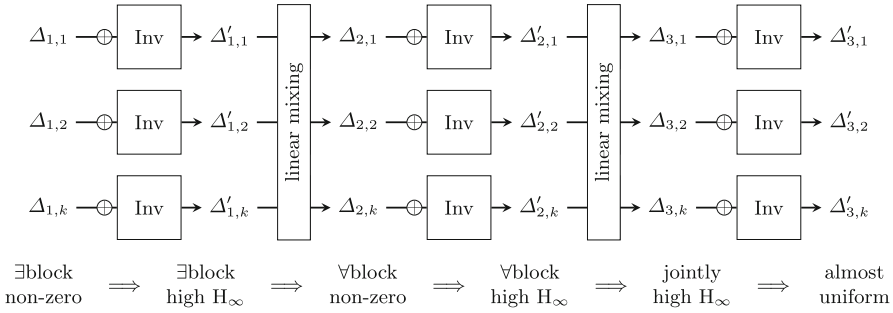


Fig. 5. Illustration of the proof of Theorem 2 and Lemma 17

Theorem 1. Assume the S-box is $P(x) = x^{2^b-2}$ over \mathbb{F}_{2^b} assume the mixing function is diffusing, that is, it satisfies Property 1. Then a 2-round SPN with k blocks each of which has b bits is ϵ -close to 2-wise independent where

$$\epsilon \leq \frac{2 + 4k}{2^b} + \sqrt{\frac{2^k - 1}{2^{b+1}}}.$$

Theorem 2. Assume the S-box is patched inverse $P(x) = x^{2^b-2}$, assume the mixing function satisfies Property 1 and Property 2. Then 3-round SPN is ϵ -close to 2-wise independent where

$$\epsilon \leq \frac{2 + 8k}{2^b} + \sqrt{\frac{k}{2^b}}.$$

Proof. Name the variables as in Fig. 5, fix any input differences $\Delta_{1,1}, \dots, \Delta_{1,k}$ which are not all zero. We wish to show that the distribution of $(\Delta'_{3,1}, \dots, \Delta'_{3,k})$ is ϵ -close to uniform. By Lemma 2, this implies ϵ -closeness to pairwise independence. We proceed via a hybrid argument.

Hybrid 0. Hybrid 0 is the real world hybrid that is illustrated in Fig. 5.

Hybrid 1. Pick some j where $\Delta_{1,j} \neq 0$. W.l.o.g., assume $\Delta_{1,1} \neq 0$. Note that the distribution of $\Delta'_{1,1}$ is $(2/2^b)$ -close to uniformly random over a subset of size 2^{b-1} (Corollary 2). Call this uniform distribution $\mathcal{D}'_{1,1}$. We have $H_\infty(\mathcal{D}'_{1,1}) = b - 1$.

Hybrid 1 is the same as hybrid 0 except that we replace $\Delta'_{1,1}$ by a random sample from the distribution $\mathcal{D}'_{1,1}$. The statistical distance from Hybrid 0 is at most $\frac{2}{2^b}$.

Claim. Assume that the mixing function satisfies Property 1. In Hybrid 1, $H_\infty[\Delta_{2,j}] \geq b - 1$ for all $j \in [k]$.

Hybrid 2. In this hybrid, we ensure $\Delta_{2,j} \neq 0$ for all $j \in [k]$. Formally, hybrid 2 is the same as hybrid 1 except that we replace $\Delta_{2,j}$ by 1 if $\Delta_{2,j} = 0$ in hybrid 1. The statistical distance from Hybrid 1 is at most $\frac{2k}{2^b}$.

Lemma 17 shows that the joint distribution of $(\Delta'_{3,1}, \dots, \Delta'_{3,k})$ is $(\frac{6k}{2^b} + \sqrt{\frac{k}{2^b}})$ close to uniform in hybrid 2.

Putting everything together, the statistical distance between $(\Delta'_{3,1}, \dots, \Delta'_{3,k})$ and the uniform distribution is at most $\frac{2+8k}{2^b} + \sqrt{\frac{k}{2^b}}$. \square

Lemma 17. *Assume the S-box is patched inverse $P(x) = x^{2^b-2}$, assume the mixing function satisfies Property 2. Starting with a pair of inputs, whose difference is entry-wise-nonzero, after a 2-round SPN, the statistical distance between the output difference and the uniform distribution is no more than $\frac{6k}{2^b} + \sqrt{\frac{k}{2^b}}$.*

Proof. Name the variables as the last two rounds in Fig. 5, fix any set of input differences $\Delta_{2,1}, \dots, \Delta_{2,k}$ which are all non-zero. We wish to show that the distribution of $(\Delta'_{3,1}, \dots, \Delta'_{3,k})$ is ϵ -close to uniform. We proceed via a hybrid argument.

Hybrid 0. Hybrid 0 is the real world hybrid.

Hybrid 1. Since $\Delta_{2,j} \neq 0$ for all $j \in [k]$, the distribution of $\Delta'_{2,j}$ is $(2/2^b)$ -close to uniformly random over a subset of size 2^{b-1} (Corollary 2). Call this uniform distribution $\mathcal{D}'_{2,j}$. We have $H_\infty(\mathcal{D}'_{2,j}) = b - 1$.

Hybrid 1 is the same as hybrid 0 except that we replace $\Delta'_{2,j}$ by a vector drawn from the distribution $\mathcal{D}'_{2,j}$ for each $j \in [k]$. The statistical distance from Hybrid 0 is at most $\frac{2k}{2^b}$.

Claim. Assume that the mixing function satisfies Property 2. In Hybrid 1, $H_\infty[\Delta_{3,j}] \geq b - 1$ for all $j \in [k]$.

Hybrid 2. In this hybrid, we change the way $\Delta'_{3,j}$ is sampled based on $\Delta_{3,j}$. In particular:

- When $\Delta_{3,j} = \delta \neq 0$, the distribution of $\pi_{\text{out}}(\Delta'_{3,j})$ conditioning on $\Delta_{3,j} = \delta$ is $\frac{2}{2^b}$ -close to uniform distribution over $\{0, \pi_{\text{in}}(\delta)\}^\perp$. Let $\pi_{\text{out}}(\Delta'_{3,j})$ sampled uniformly from $\{0, \pi_{\text{in}}(\delta)\}^\perp$ in hybrid 2.
- When $\Delta_{3,j} = 0$, $\Delta'_{3,j}$ is chosen to be uniformly random in hybrid 2.

Let us calculate the statistical distance between hybrids 1 and 2. The first bullet introduces a statistical distance of at most $2k/2^b$. The probability that a fixed coordinate $\Delta_{3,j}$ is 0 is at most $2/2^b$, and therefore, the probability that some coordinate is 0 is at most $2k/2^b$. In total, the statistical distance is at most $\frac{4k}{2^b}$.

By applying our extraction lemma⁸ (Lemma 11), we know that, in hybrid 2, the joint distribution of $\Delta'_{3,1}, \dots, \Delta'_{3,k}$ is at most $\sqrt{\frac{k}{2^b}}$ -away from uniform.

Counting them together, the statistical distance between $(\Delta'_{3,1}, \dots, \Delta'_{3,k})$ and the uniform distribution is at most $\frac{6k}{2^b} + \sqrt{\frac{k}{2^b}}$. □

3.4 AES is Almost Pairwise-Independent

Good asymptotic bounds have been shown in Theorem 1 and 2, but the analysis there is way too loose on AES parameter ($k = 16, b = 8$). This section emphasizes on better concrete bound. Comparing with Sect. 3.3, the concrete bound is improved by the following tricks.

- Lemma 11 shows that the statistical distance is less than $\frac{1}{2} \cdot \sqrt{\left(1 + \frac{1}{2^h}\right)^k - 1}$, which is less than $\sqrt{\frac{k}{2^{h+1}}}$. The former is tighter. In particular, when $k = 16, b = 8, h = -\log_2\left(\frac{2}{2^b} + \frac{8}{2^{2b}}\right)$, the former shows $d_{TV} \leq 0.18357\dots \leq \frac{47}{256}$, and the latter shows $d_{TV} \leq 0.25$.
- Lemma 18 is the strengthening of Lemma 17. Besides using the tighter bound from Lemma 11, it also considers Rényi entropy instead of min-entropy.
- Theorem 3 is the strengthening of Theorem 2. The proof of Theorem 3 (resp. Theorem 2) shows that after two rounds of AES (resp. one round of SPN), all block differences are non-zero with high probability. Then ignoring the rare event, Lemma 18 (resp. Lemma 17) will conclude the proof. The proof of Theorem 3 also carefully analyzes the rare event that some block difference is zero after 2 rounds of AES. It observes that, given the rare event happens, after two more rounds, all block differences will be non-zero with high probability.

Lemma 18 (Strengthening of Lemma 17). *Assume the S-box is patched inverse $P(x) = x^{2^b-2}$, assume the mixing function satisfies Property 2. Starting with a pair of inputs, whose difference is entry-wise-nonzero, after a 2-round SPN, the statistical distance between the output difference and the uniform distribution is no more than $\frac{4k}{2^b} + \frac{1}{2} \sqrt{\left(1 + 2^{-h}\right)^k - 1}$, where $h = -\log_2\left(\frac{2}{2^b} + \frac{8}{2^{2b}}\right)$.*

In particular, when $k = 16, b = 8$, we have $d_{TV} \leq \frac{64+47}{256}$.

The proof is mostly the same of Lemma 17 and is deferred to the full version [40].

Lemma 19. *Starting with a pair of distinct inputs, after 2-round of AES, including a tailing linear mixing, the output difference has zero entry with probability no more than $\frac{25}{27}$.*

Theorem 3. *6-round of AES is 0.472-close to pairwise independence.*

The proof is similar to that of Theorem 2 and is deferred to the full version [40].

⁸ Our extraction lemma also requires $k \leq 2^{b-1}$. In the case $k > 2^{b-1}$, Lemma 17 can be trivially proved as $d_{TV} \leq 1 \leq \frac{6k}{2^b} + \sqrt{\frac{k}{2^b}}$.

3.5 Multi-round SPNs and AES

We now combine the bounds from Theorems 1, 2, and 3 with the MPR amplification lemma (Lemma 1) to obtain the following theorems.

Theorem 4. *Assume the S-box is $P(x) = x^{2^b-2}$ over \mathbb{F}_{2^b} assume the mixing function is diffusing, that is, it satisfies Property 1. Then a $(2r)$ -round SPN with k blocks each of which has b bits is ϵ -close to 2-wise independent where*

$$\epsilon \leq 2^{r-1} \left(\frac{2 + 4k}{2^b} + \sqrt{\frac{2^k - 1}{2^{b+1}}} \right)^r .$$

Further, if the mixing function additionally satisfies Property 2, then $(3r)$ -round SPN is ϵ -close to 2-wise independent where

$$\epsilon \leq 2^{r-1} \left(\frac{2 + 8k}{2^b} + \sqrt{\frac{k}{2^b}} \right)^r .$$

Theorem 5. *6r-round AES is $2^{r-1}(0.472)^r$ -close to pairwise independence.*

4 t -wise Independence of KAC

In this section, we consider a key-alternating cipher whose i^{th} round consists of applying a *public, fixed* permutation p_i to the current state followed by adding a (private) round-key s_i . The main result of this section is that for every r , there exist public permutations p_1, \dots, p_r such that r rounds of KAC using these permutations gets us close to $(r - o(r))$ -wise independence. We achieve a strong notion of *pointwise* closeness (see Definition 6) much stronger than the statistical distance measures considered in previous sections. Furthermore, it is easy to see that a t -round KAC can at best be (close to) t -wise independence, due to a simple entropy argument, meaning that our result is nearly optimal and entropy-preserving.

We remark that this is an *existential result*: namely, we do not explicitly construct the fixed permutations used by the KAC, but merely show that they exist. Indeed, we show that most permutations work, as is typical of probabilistic arguments. We also remark that the permutations p_1, \dots, p_r are fixed and known to the adversary, thus the only secret randomness in the construction comes from the round keys s_i .

We start with some new notations. We encourage the reader to consult the full version [40] for tail bounds that are extensively used in our analysis.

4.1 Definitions and Notations

Let \mathfrak{D} denote the domain and let $2^n = N := |\mathfrak{D}|$. Throughout this report, we will consider many distribution of permutations over \mathfrak{D} . Permutation distributions will be denoted by calligraphic letters (e.g. $\mathcal{F}, \mathcal{G}, \mathcal{H}$). A random choice of a permutation from such a distribution will act as a key for the KAC. Here are two simple examples of permutation distributions:

Example 1 (Shift permutations). Denoted by \mathcal{S} , the uniform distribution over

$$\{\sigma_s : x \mapsto x + s \mid s \in \mathfrak{D}\},$$

which consists of all shift permutations σ_s that additively shifts the input by s . The definition assumes \mathfrak{D} to be a group. The support of \mathcal{S} is of size N .

We now define a notation for composition of permutations, the cornerstone of the KAC construction.

Definition 5 (Composition). Let \mathcal{F}, \mathcal{G} be distributions over permutations, and let p be a permutation over \mathfrak{D} . Their compositions are defined as

- $\mathcal{F} \circ p$ is the distribution of $f \circ p$ where $f \leftarrow \mathcal{F}$,
- $p \circ \mathcal{G}$ is the distribution of $p \circ g$ where $g \leftarrow \mathcal{G}$,
- $\mathcal{F} \circ \mathcal{G}$ is the distribution of $f \circ g$ where $f \leftarrow \mathcal{F}, g \leftarrow \mathcal{G}$ independently.

Key Alternating Cipher. Given the language of permutation distributions from above, we can give an alternative definition of key-alternating ciphers (KACs). A t -round KAC is parametered by fixed permutations p_1, \dots, p_{t-1} , and is the composition

$$\mathcal{S} \circ p_1 \circ \mathcal{S} \circ p_2 \circ \mathcal{S} \circ p_3 \circ \dots \circ p_{t-1} \circ \mathcal{S}.$$

In words, this means picking t round-keys $s_1, \dots, s_t \leftarrow \mathfrak{D}$ and letting

$$f_{s_1, \dots, s_t}(x) = s_t + \underbrace{p_{t-1}(s_{t-1} + p_{t-2}(s_{t-2} + \dots))}_{\text{repeated } t - 1 \text{ times}}$$

as illustrated in Fig. 1.

Pointwise Closeness to t -wise Independence. Finally, we define the notion of being pointwise close to t -wise independent which we achieve. It is a stronger notion than being close to t -wise independent (Definition 3), a notion that we worked with in Sect. 4. This only makes the results of this section stronger.

Definition 6 (pointwise close to t -wise independence). Let \mathcal{F} be a distribution over permutations. \mathcal{F} is pointwise ϵ -close to t -wise independence if for any distinct $x_1, \dots, x_t \in \mathfrak{D}$ and any distinct $y_1, \dots, y_t \in \mathfrak{D}$,

$$\Pr_{f \leftarrow \mathcal{F}} \left[f(x_1) = y_1 \wedge f(x_2) = y_2 \wedge \dots \wedge f(x_t) = y_t \right] \in \left(\frac{1 - \epsilon}{N^t}, \frac{1 + \epsilon}{N^t} \right).$$

4.2 Existential Results for Key Alternating Ciphers

In this section, we will prove our main existential result, that is, for some $r = t + o(t) + s$, there exist permutations p_1, \dots, p_r such that a r -round KAC using these permutations is $\exp(-s)$ -close to t -wise independent.

The result is proved by a careful induction that combines two steps.

- *Independence Amplification:* Lemma 20 shows that if \mathcal{F} is pointwise ε -close to t -wise independent, then $\mathcal{S} \circ p \circ \mathcal{F}$ is pointwise $(c(1+\varepsilon)t^2 \log N)$ -close to $(t+1)$ -wise independent, for most permutations p and for some constant $c > 1$. In other words, one more KAC round takes you from *very t -wise independent* to *somewhat $(t+1)$ -wise independent*. It is important to note that even though the distance of the resulting permutation is $c(1+\varepsilon)t^2 \log N \gg 1$, this is still a non-trivial pointwise guarantee. In fact, one can inductively apply Lemma 20 and conclude that t -round KAC is pointwise $((t!)^2(c \log N)^{t-1})$ -close to t -wise independence, starting from just 1-wise independence. As mentioned before, although the distance is much larger than 1, this is a non-trivial statement, because it is about pointwise closeness.
- *Distance Amplification:* Lemma 21 will reduce the distance to t -wise independence by adding more rounds. Say \mathcal{F} is pointwise ε -close to t -wise independent and is pointwise ε' -close to $(t+1)$ -wise independent, where $\varepsilon' \gg \varepsilon$. I.e., \mathcal{F} is very close to t -wise independent and somewhat close to $(t+1)$ -wise independent. Lemma 21 shows that adding one more round makes it much closer to $(t+1)$ -wise independent. More formally, $\mathcal{S} \circ p \circ \mathcal{F}$ is pointwise $(\varepsilon + \tilde{O}(\frac{\varepsilon' t}{\sqrt[3]{N}}))$ -close to $(t+1)$ -wise independent, for most permutations p .

Iterated applications of Lemmas 20 and 21 takes us very close to t -wise independence in $2t$ rounds. Indeed, it is not hard to see that one can do even better: between any two successive applications of distance amplification, one can afford to do a large number ($\approx \log N / \log \log N$ many) of iterations of independence amplification. Therefore, to get to t -wise independence, it suffices to work with a $(t + o(t))$ -round KAC.

For example, 1-round KAC is 1-wise independent. Then, 2-round KAC is $O(\log N)$ -close to 2-wise independent, due to Lemma 20. By adding one more round, Lemma 21 shows that 3-round KAC is $O(\frac{\log N}{N})$ -close to 2-wise independent. Figure 6 illustrates the progression of the inductive argument.

More generally, we show:

Theorem 6 (Main KAC Theorem). *For every t , let $r = t + o(t)$. There exist fixed permutations p_1, \dots, p_r such that the r -round key-alternating cipher is $1/N^{\Omega(1)}$ -close to t -wise independent.*

The theorem follows from Lemma 20 and Lemma 21 below whose proofs are deferred to the full version [40]. Finally, we remark that the proof of the theorem shows more: that an overwhelming fraction of choices of permutations p_1, \dots, p_r gives us a t -wise independent KAC.

Lemma 20. *Let \mathcal{F} be a distribution which is pointwise ε -close to ℓ -wise independence. At least $1 - 1/N^{t+1}$ of the possible permutations p satisfy the property that $\mathcal{S} \circ p \circ \mathcal{F}$ is pointwise $O((1+\varepsilon)(t+1)^2 \log N)$ -close to $(t+1)$ -wise independence.*

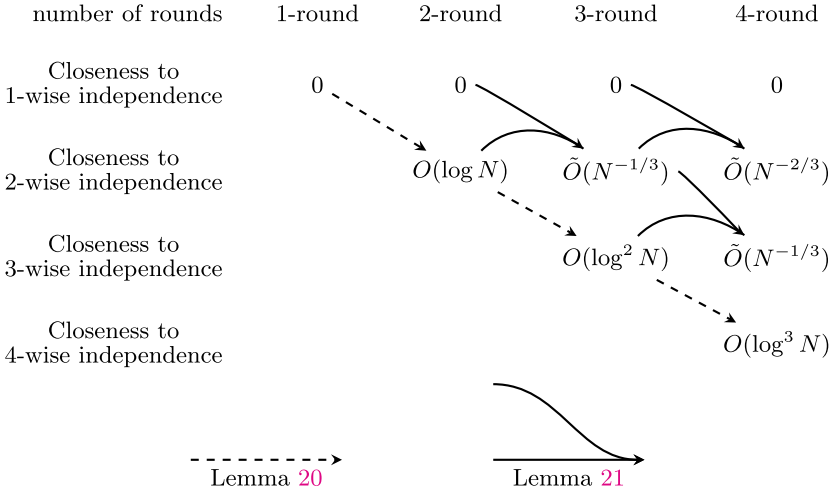


Fig. 6. Illustration of the inductive proof using Lemmas 20, 21.

Lemma 21. *Let \mathcal{F} be a permutation distribution that is pointwise ε -close to t -wise independence and is pointwise ε' -close to $(t + 1)$ -wise independence. At least $1 - 1/N^{t+1}$ of the possible permutations p satisfy the property that $\mathcal{S} \circ p \circ \mathcal{F}$ is pointwise $(\varepsilon + 4\varepsilon'(t + 1) \sqrt[3]{\ln N/N})$ -close to $(t + 1)$ -wise independence.*

Acknowledgments. The third author thanks Charlie Rackoff for inspiring discussions a decade ago on the topic of what can be proved about the security of block ciphers. He also thanks Orr Dunkelman and Thomas Peyrin for answering his questions about the role of key schedule in the security of AES. We thank the Simons Institute for hosting us at the “Lattices: Algorithms, Complexity and Cryptography” program where the seeds of this work were planted. TL was supported by NSF grants CNS-1528178, CNS-1929901, CNS-1936825 (CAREER), CNS-2026774, a JP Morgan AI research Award, and a Simons Foundation Collaboration Grant on Algorithmic Fairness. ST was supported in part by NSF grants CNS-1930117 (CAREER), CNS-1926324, CNS-2026774, a Sloan Research Fellowship, and a JP Morgan Faculty Award. VV was supported by DARPA under Agreement No. HR00112020023, a grant from the MIT-IBM Watson AI, a grant from Analog Devices, a Microsoft Trustworthy AI grant, and a DARPA Young Faculty Award.

References

1. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: MiMC: efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 191–219. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_7
2. Alon, N., Lovett, S.: Almost k -wise vs. k -wise independent permutations, and uniformity for general group actions. *Theory Comput.* **9**, 559–577 (2013)
3. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indistinguishability of key-alternating ciphers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 531–550. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_29

4. Baignères, T., Vaudenay, S.: Proving the security of AES substitution-permutation network. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 65–81. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_5
5. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **4**(1), 3–72 (1991)
6. Blondeau, C., Nyberg, K.: Perfect nonlinear functions and cryptography. *Finite Fields Their Appl.* **32**, 120–147 (2015)
7. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_19
8. Bogdanov, A., Knudsen, L.R., Leander, G., Standaert, F.-X., Steinberger, J., Tischhauser, E.: Key-alternating ciphers in a provable setting: encryption using a small number of public permutations. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 45–62. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_5
9. Brodsky, A., Hoory, S.: Simple permutations mix even better. *Random Struct. Algorithms* **32**(3), 274–289 (2008)
10. Caranti, A., Volta, F.D., Sala, M.: An application of the O’Nan-Scott theorem to the group generated by the round functions of an AES-like cipher. *Des. Codes Cryptogr.* **52**(3), 293–301 (2009)
11. Chen, S., Lampe, R., Lee, J., Seurin, Y., Steinberger, J.P.: Minimizing the two-round Even-Mansour cipher. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 39–56. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_3
12. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_19
13. Cogliati, B., et al.: Provable Security of (Tweakable) Block Ciphers Based on Substitution-Permutation Networks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 722–753. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_24
14. Cogliati, B., Lee, J.: Wide tweakable block ciphers based on substitution-permutation networks: security beyond the birthday bound. *IACR Cryptol. ePrint Arch.* **2018**, 488 (2018)
15. Coppersmith, D., Grossman, E.: Generators for certain alternating groups with applications to cryptography. *SIAM J. Appl. Math.* **29**(4), 624–627 (1975)
16. Daemen, J.: Cipher and hash function design strategies based on linear and differential cryptanalysis. Ph.D. thesis, KU Leuven (1995)
17. Daemen, J., Rijmen, V.: Understanding two-round differentials in AES. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 78–94. Springer, Heidelberg (2006). https://doi.org/10.1007/11832072_6
18. Dodis, Y., Katz, J., Steinberger, J.P., Thiruvengadam, A., Zhang, Z.: Provable security of substitution-permutation networks. *IACR Cryptol. ePrint Arch.* **2017**, 16 (2017)
19. Dodis, Y., Stam, M., Steinberger, J.P., Liu, T.: Indifferentiability of confusion-diffusion networks. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 679–704. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_24

20. Guo, C., Lin, D.: On the indifferentiability of key-alternating feistel ciphers with no key derivation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 110–133. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_6
21. Guo, C., Lin, D.: A synthetic indifferentiability analysis of interleaved double-key Even-Mansour ciphers. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 389–410. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_16
22. Hoang, V.T., Tessaro, S.: Key-alternating ciphers and key-length extension: exact bounds and multi-user security. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 3–32. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_1
23. Hong, S., Lee, S., Lim, J., Sung, J., Cheon, D., Cho, I.: Provable security against differential and linear cryptanalysis for the SPN structure. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 273–283. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44706-7_19
24. Hoory, S., Magen, A., Myers, S.A., Rackoff, C.: Simple permutations mix well. *Theor. Comput. Sci.* **348**(2–3), 251–261 (2005)
25. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 28–40. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052332>
26. Kaliski, B.S., Jr., Rivest, R.L., Sherman, A.T.: Is the data encryption standard a group? (results of cycling experiments on DES). *J. Cryptol.* **1**(1), 3–36 (1988)
27. Kang, J.-S., Hong, S., Lee, S., Yi, O., Park, C., Lim, J.: Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks. *ETRI J.* **23**, 02 (2002)
28. Kaplan, E., Naor, M., Reingold, O.: Derandomized constructions of k -wise (almost) independent permutations. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX/RANDOM 2005. LNCS, vol. 3624, pp. 354–365. Springer, Heidelberg (2005). https://doi.org/10.1007/11538462_30
29. Keliher, L.: Refined analysis of bounds related to linear and differential cryptanalysis for the AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2004. LNCS, vol. 3373, pp. 42–57. Springer, Heidelberg (2005). https://doi.org/10.1007/11506447_5
30. Keliher, L., Meijer, H., Tavares, S.: Improving the upper bound on the maximum average linear hull probability for rijndael. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 112–128. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45537-X_9
31. Keliher, L., Meijer, H., Tavares, S.: New method for upper bounding the maximum average linear hull probability for SPNs. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 420–436. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_26
32. Keliher, L., Sui, J.: Exact maximum expected differential and linear probability for two-round advanced encryption standard. *IET Inf. Secur.* **1**(2), 53–57 (2007)
33. Knudsen, L.: Deal - a 128-bit block cipher. In: NIST AES Proposal (1998)
34. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_16
35. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45661-9_9

36. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) *Communications and Cryptography*, pp. 227–233. Springer, Boston (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
37. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_2
38. Lampe, R., Seurin, Y.: Security analysis of key-alternating feistel ciphers. In: Cid, C., Rechberger, C. (eds.) *FSE 2014*. LNCS, vol. 8540, pp. 243–264. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_13
39. Lidl, R., Niederreiter, H.: *Introduction to Finite Fields and Their Applications*. Cambridge University Press, Cambridge (1986)
40. Liu, T., Tessaro, S., Vaikuntanathan, V.: The t -wise independence of substitution-permutation networks. *IACR Cryptol. ePrint Arch.* **2021**, 507 (2021)
41. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) *EUROCRYPT 1992*. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-47555-9_7
42. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) *CRYPTO 2007*. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_8
43. Miles, E., Viola, E.: Substitution-permutation networks, pseudorandom functions, and natural proofs. *J. ACM* **62**(6), 46:1–46:29 (2015)
44. Murphy, S., Paterson, K.G., Wild, P.R.: A weak cipher that generates the symmetric group. *J. Cryptol.* **7**(1), 61–65 (1994)
45. Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_6
46. Nyberg, K., Knudsen, L.R.: Provable security against a differential attack. *J. Cryptol.* **8**(1), 27–37 (1995)
47. O’Donnell, R.: *Analysis of Boolean Functions*. Cambridge University Press, Cambridge (2014)
48. Park, S., Sung, S.H., Chee, S., Yoon, E.-J., Lim, J.: On the security of rijndael-like structures against differential and linear cryptanalysis. In: Zheng, Y. (ed.) *ASIACRYPT 2002*. LNCS, vol. 2501, pp. 176–191. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_11
49. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In: Johansson, T. (ed.) *FSE 2003*. LNCS, vol. 2887, pp. 247–260. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39887-5_19
50. Steinberger, J.P.: Improved security bounds for key-alternating ciphers via hellinger distance. *IACR Cryptol. ePrint Arch.* **2012**, 481 (2012)
51. Vaudenay, S.: Decorrelation: a theory for block cipher security. *J. Cryptol.* **16**(4), 249–286 (2003)
52. Wu, Y., Yu, L., Cao, Z., Dong, X.: Tight security analysis of 3-round key-alternating cipher with a single permutation. In: Moriai, S., Wang, H. (eds.) *ASIACRYPT 2020*. LNCS, vol. 12491, pp. 662–693. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_22

Low-Complexity Cryptography



Low-Complexity Weak Pseudorandom Functions in $\text{AC0}[\text{MOD2}]$

Elette Boyle^{1(✉)}, Geoffroy Couteau², Niv Gilboa³, Yuval Ishai⁴, Lisa Kohl⁵,
and Peter Scholl⁶

¹ IDC Herzliya, Herzliya, Israel
eboyle@alum.mit.edu

² IRIF, Paris, France
couteau@irif.fr

³ Ben-Gurion University, Beersheba, Israel
gilboan@bgu.ac.il

⁴ Technion, Haifa, Israel

yuvali@cs.technion.ac.il

⁵ Cryptology Group, CWI Amsterdam, Amsterdam, The Netherlands
lisa.kohl@cwi.nl

⁶ Aarhus University, Aarhus, Denmark
peter.scholl@cs.au.dk

Abstract. A *weak pseudorandom function* (WPRF) is a keyed function $f_k : \{0, 1\}^n \rightarrow \{0, 1\}$ such that, for a random key k , a collection of samples $(x, f_k(x))$, for *uniformly random* inputs x , cannot be efficiently distinguished from totally random input-output pairs (x, y) . We study WPRFs in $\text{AC0}[\text{MOD2}]$, the class of functions computable by AC0 circuits with parity gates, making the following contributions.

- **WPRF by sparse polynomials.** We propose the first WPRF candidate that can be computed by sparse multivariate polynomials over \mathbb{F}_2 . We prove that it has subexponential security against linear and algebraic attacks.
- **WPRF in $\text{AC0} \circ \text{MOD2}$.** We study the existence of WPRFs computed by AC0 circuits *over* parity gates. We propose a modified version of a previous WPRF candidate of Akavia et al. (ITCS 2014), and prove that it resists the algebraic attacks that were used by Bogdanov and Rosen (ECCC 2017) to break the original candidate in quasipolynomial time. We give evidence against the possibility of using *public* parity gates and relate this question to other conjectures.
- **Between Lapland and Cryptomania.** We show that WPRFs in $\text{AC0}[\text{MOD2}]$ imply a variant of the Learning Parity with Noise (LPN) assumption. We further show that WPRFs in a subclass of $\text{AC0}[\text{MOD2}]$ that includes a recent candidate by Boyle et al. (FOCS 2020) imply, under a seemingly weak additional conjecture, public-key encryption.

1 Introduction

This work explores the minimal achievable complexity of *weak pseudorandom functions*. Roughly speaking, a pseudorandom function (PRF) family [31] is a collection of efficiently computable functions $f_k(x)$, such that a random function from

the collection induced by a uniform choice of the key k cannot be efficiently distinguished from a truly random function. The existence (or nonexistence) of PRFs in low complexity classes is closely related to questions in computational learning theory [38, 58]: Indeed, any complexity class rich enough to contain PRFs is inherently unlearnable, even when membership queries are allowed. In this light, understanding the feasibility of low-complexity PRFs corresponds to exploring the border between the learnable and the unlearnable. More broadly, the study of low-complexity PRFs has proven to be a rich and fruitful research direction, motivated by many connections with circuit lower bounds [43, 54, 56], derandomization [49, 61], and “high-end” cryptographic applications [2, 8, 14, 15, 17, 42].

We focus on the existence of *weak* pseudorandom functions (WPRFs) in $\text{AC0}[\text{MOD2}]$, the class of polynomial-size, constant-depth circuits over AND, OR, XOR gates and negations.¹

Informally, a WPRF relaxes a PRF by restricting the distinguisher to only get input-output pairs for *uniformly random* inputs x , as opposed to chosen inputs x . WPRFs imply hardness results for learning (without membership queries) under the uniform distribution, and can serve as useful building blocks for most “symmetric” cryptographic primitives, such as private-key encryption and message authentication [46]. As a result, minimizing their complexity can lead to improving the complexity of these primitives.

Levels of security. We say that a WPRF has quasipolynomial, subexponential, or exponential security when the distinguisher’s circuit size is bounded by a corresponding function of the key length. Concretely, there exists $c > 0$ such that every circuit of size $T = n^{\log^c n}$, $T = 2^{n^c}$, or $T = 2^{cn}$ (respectively) has at most $1/T$ distinguishing advantage between f_k and a random function, for all sufficiently large key lengths n , given unlimited access to examples on uniformly random inputs. In the case of quasipolynomial and subexponential security, we can equivalently let n be the input length, since the key length and input length are polynomially related. In this work we consider subexponential security by default. This is typically the best level of security achieved by constructions from standard cryptographic assumptions.

WPRFs in low complexity classes. We return to the question of WPRFs in $\text{AC0}[\text{MOD2}]$. At the lower end, much is known about the power and limitations of AC0 . This includes unconditional circuit lower bounds (e.g. AC0 cannot compute parity [28, 32]), derandomization (e.g. AC0 cannot distinguish any polylog-wise independent distribution from the uniform distribution [16]), and learning algorithms (e.g. AC0 can be learned from quasipolynomially many samples under the uniform distribution [41]). The latter imply, in particular, that AC0 cannot contain a WPRF with better than quasipolynomial security. Slightly above

¹ More precisely, AND/OR/XOR gates can have an unbounded fan-in, and depth is defined to be the length of the longest path from an input to the output, not counting negations. As is common in the study of constant-depth PRFs, we consider the complexity of mapping the input to the output when the key is fixed.

$\text{AC0}[\text{MOD2}]$, the picture is also relatively clear: strong PRFs with subexponential security exist in the class TC0 (of polynomial-size constant-depth circuits with threshold gates) under standard cryptographic assumptions [9, 47, 48]. In contrast, despite some partial results [1, 6, 14, 15, 60], the space in between AC0 and TC0 remains a relatively uncharted territory.

Sparse \mathbb{F}_2 -polynomials. Sparse polynomials are a natural object of study in several areas, including computational learning theory. We will be interested in sparse n -variate polynomials over \mathbb{F}_2 , namely sums of $\text{poly}(n)$ monomials. Sparse \mathbb{F}_2 -polynomials can be viewed as the subclass of $\text{AC0}[\text{MOD2}]$ corresponding to depth-2 circuits that take the XOR of ANDs of inputs. A WPRF in this class would show the hardness of learning sparse \mathbb{F}_2 -polynomials under the uniform distribution. We briefly survey some relevant known results.

A result of Hellerstein and Servedio [34] implies an $2^{\tilde{O}(\sqrt{n})}$ -time PAC learning algorithm (applying to any input distribution) for learning sparse \mathbb{F}_2 -polynomials. In the converse direction, a recent work of Daniely and Vardi [24] shows that sparse \mathbb{F}_2 -polynomials are hard to learn in better than *quasipolynomial time*, albeit only under a specific non-uniform input distribution (a highly biased Bernoulli distribution), under the conjectured existence of polynomial-stretch local pseudorandom generators [4, 30, 35]. Finally, Boneh et al. [14] put forward a WPRF candidate in ACC0 that implies $2^{\Omega(n)}$ -hardness of learning sparse \mathbb{F}_3 -polynomials, again under a special input distribution (uniform over $\{-1, 1\}^n$). To our knowledge, no result is currently known that supports the hardness of learning sparse \mathbb{F}_2 -polynomials in *any* hardness regime under the uniform distribution, or in the subexponential hardness regime under *any* distribution.

The class $\text{AC0} \circ \text{MOD2}$. The class $\text{AC0} \circ \text{MOD2}$ of AC0 on *top* of parities can be seen as a minimal extension of AC0. Despite its apparent simplicity, it is quite poorly understood. In particular, it is open whether the mod-2 inner-product function is in this class [56]. Akavia et al. [1] put forward the question of WPRFs in $\text{AC0} \circ \text{MOD2}$ as a second-best alternative to WPRFs in AC0. They presented a candidate construction where $f_k(x)$ applies a specific DNF formula (the “TRIBES” function) to a secret linear mapping $A_k \cdot x$ of the input x , and proved resistance against several classes of attacks. However, this candidate was later broken by a quasipolynomial-time algebraic attack [13] exploiting the low *rational degree* of functions f_k in the family. Namely, there exists a low-degree g for which $f_k \cdot g = 0$ or $(f_k \oplus 1) \cdot g = 0$. This kind of attacks further rules out the possibility of any WPRF with better than quasipolynomial security that can be computed by depth-2 AC0 circuits over XOR.

1.1 Our Contribution

Candidate WPRF by sparse \mathbb{F}_2 -polynomials. We present a candidate WPRF in the class of sparse \mathbb{F}_2 -polynomials that can be conjectured to have subexponential security. More concretely, we conjecture our candidate to be secure against distinguishers of size $T = 2^{n^\epsilon}$ for a constant $\epsilon \geq 1/8$, where

n is the input size. To our knowledge, this is the first proposal for a candidate WPRF in this class. We give several kinds of evidence for the security of our candidate. First, building on previous works, we show that it has high rational degree. This implies that it cannot be broken by a subexponential algebraic attack (the same attack that breaks the candidate of [1] in quasipolynomial time). Second, we reduce its security to a *variable-density* variant of the Learning Parity with Noise (LPN) [12] assumption. This assumption is similar to (but essentially incomparable) to the variable-density LPN assumption used in the recent work of Boyle et al. [15] to build a WPRF in the class of XNF formulas (sparse \mathbb{F}_2 -polynomials in the inputs *and their negations*). Finally, we prove that it cannot be broken by any attack that fits into the framework of *linear attacks*, a general framework that captures in particular all known attacks against the LPN assumption and its variants. Our analysis builds upon the analysis of [15]; however, our setting involves additional challenges that require to significantly refine their proof techniques.

Our candidate WPRF provides an explicit distribution \mathcal{D} over sparse n -variate \mathbb{F}_2 -polynomials such that the following plausibly holds: no circuit of size $2^{n^{1/s}}$, given the values of a secret polynomial $p \in_R \mathcal{D}$ on uniformly random inputs, can predict the value of p on a fresh random input with better than $2^{-n^{1/s}}$ advantage.

As noted above, the recent work of Daniely and Vardi [24] shows hardness of learning sparse \mathbb{F}_2 -polynomials, assuming the existence of local pseudorandom generators. Our results are incomparable (and complementary) to their result:

- The result of [24] only shows the hardness of learning sparse \mathbb{F}_2 -polynomials for inputs sampled from a very specific distribution \mathcal{D} over strings $\{0, 1\}^n$, which outputs n independent samples from a highly biased Bernoulli distribution. In contrast, our results hold with respect to the *uniform* distribution.
- The result of [24] fundamentally cannot apply to the subexponential regime. The core reason is the following: from the existence of a learner for s -sparse polynomials given N examples, [24] only derives a contradiction to the existence of $(\log s)$ -local PRGs which stretch N bits from their input. However, it is known [44] that logarithmic-locality pseudorandom generators cannot possibly achieve stretch beyond quasipolynomial. Therefore, their result does not apply to the setting where s is polynomial and N is subexponential. In contrast, our result applies even to subexponential-time learning algorithms, in the setting where s is polynomial.
- On the other hand, the result of [24] relies on the existence of local PRGs, which is a relatively well-established assumption. In contrast, our result relies on a new variant of LPN, which we support by proving that it resists a large class of attacks (including in particular all standard attacks against LPN).

Candidate weak PRF in $\text{AC0} \circ \text{MOD2}$. We revisit the question of Akavia et al. [1]:

Can weak pseudorandom functions exist in the class $\text{AC0} \circ \text{MOD2}$?

We present a new candidate WPRF in $\text{AC0} \circ \text{MOD2}$ which follows the high-level template of Akavia et al. [1], but with an alternative choice of AC0 circuit structure. The WPRF candidate of Akavia et al. [1] (hereafter referred to as the “ABGKR” candidate) is of the form

$$f_{s,K}(x) = \langle x, s \rangle \oplus g(K \cdot x \pmod 2)$$

for $s \in \{0,1\}^n$, $K \in \{0,1\}^{(n-1) \times n}$, where $g(x) = \bigvee_{i=1}^\lambda \bigwedge_{j=1}^{\log \lambda} x_{ij}$ is a DNF (the so-called TRIBES function). Since $f_{s,K}(x)$ can be written as $(\neg \langle x, s \rangle \wedge g(K \cdot x)) \vee (\langle x, s \rangle \wedge \neg g(K \cdot x))$, it indeed belongs to $\text{AC0} \circ \text{MOD2}$. Notice that this candidate is an instance of the learning parity with simple deterministic noise framework, where $g(\cdot)$ is the noise function. Since the noise function is biased, XORing it with $\langle x, s \rangle$ makes the final function balanced.

Unfortunately, this candidate was broken in [13] by an algebraic attack. In our candidate, we address this issue by simply adding a layer of OR gates after the parity layer, replacing the noise function with:

$$g(x) = \bigvee_{i=1}^\lambda \bigwedge_{j=1}^\lambda \bigvee_{k=1}^w x_{ijk}.$$

We conjecture that our candidate is a subexponentially secure WPRF. We observe that our candidate resists the same classes of attacks as addressed for the ABGKR candidate. However, we are further able to prove that our candidate construction has *high rational degree*, thus circumventing the algebraic attacks under which the ABGKR candidate was insecure.²

We also study the resistance of our candidate against *linear attacks*, a large class of attacks that includes most state-of-the-art attacks on learning parity problems (such as the learning parity with noise assumption), whose structure bears connections to our candidate. We put forth a conjecture which, if true, implies that our candidate (as well as the WPRF candidates of [1, 14]) cannot be broken by linear attacks.

We view our results as providing a strong indication that $\text{AC0} \circ \text{MOD2}$ may not be learnable under the uniform distribution. We compare our results to known results regarding low-complexity PRFs on Table 1. As shown in the Table, our work fills gaps in our understanding of the complexity of weak PRFs.

On WPRFs in AC0 on top of *public* parities. The conjectured security of our candidate above relies on the MOD2 portion of the $\text{AC0} \circ \text{MOD2}$ circuit remaining *secret*, dictated by the secret WPRF key. We further revisit the question:

Can WPRF exist in the class formed by AC0 atop public parities?

² Formally, high rational degree does not prove resistance to the attack from [13], which only requires *proximity* to low rational degree. However, we view this as strong evidence that the attack does not apply to our candidate.

Table 1. Comparison of positive and negative results for low-depth PRFs. We consider the complexity of computing the output for any fixed key, where security level is with respect to the key length (see Definition 4). We write $\text{AC0}[\text{MOD}2]$ to denote the class AC0 with XOR gates at all levels, and ACC0 to denote the class AC0 with MOD_m gates for a fixed integer m ($m = 6$ suffices). RLF refers to the conjectured one-wayness of random local functions [30] and Factor to the intractability of factoring.

Circuit Class	Reference	Flavor	Security	Assumption
AC0	[12]	Weak PRF	Quasipolynomial	Heuristic
	[6, 39]	Weak PRF	Quasipolynomial	Factor, RLF
	[41]	No WPRF with better than quasipolynomial sec.		
AC0 + O(1) XOR, MAJ	[60]	No Strong PRF		
Sparse \mathbb{F}_2 -polynomials	This work	Weak PRF	Subexponential	Heuristic
	[34]	No WPRF with input length n and better than $2^{\tilde{O}(\sqrt{n})}$ sec.		
XNF formulas	[15]	Weak PRF	Subexponential	Heuristic
AC0 \circ MOD2	[1, 13]	Weak PRF	Quasipolynomial	Heuristic
	This work	Weak PRF	Subexponential	Heuristic
AC0[MOD2]	[39, 47, 60]	Strong PRF	Quasipolynomial	DDH, Factor
	[15]	Weak PRF	Subexponential	Heuristic
	[18, 40, 54]	No strong PRF with better than quasipolynomial sec.		
ACC0	[14]	Weak & Strong	Exponential	Heuristic
almost-AC0[MOD2]	[62]	Strong PRF	Subexponential	Low-noise LPN
quasilinear-TC0	[43]	Strong PRF	Exponential	Heuristic
TC0	[9, 47, 48]	Strong PRF	Subexponential	LWE, DDH, Fact

That is, we study the (in)existence of WPRFs of the form $f_k(x) = g_k(G \cdot x)$, where $g_k \in \text{AC0}$ and G is a public matrix. The existence of such a candidate would imply AC0 is not weakly learnable on all linear distributions (i.e. uniform distributions over linear subspaces of \mathbb{F}_2^n). Note, however, that it does not directly imply strong learnability, as boosting techniques would require the learner to modify the input distribution, an option that is not available for WPRFs.

We put forth a conjecture regarding the heavy Fourier coefficients of functions of this form, which implies that no WPRF can exist in AC0 on top of public parities. This is a direct strengthening of a conjecture of [1], which asserts the *existence* of a heavy Fourier coefficient for any function in this class. We conjecture further about the *form* of a heavy Fourier coefficient: namely, its expressibility as $G^T \cdot b$ for a low-weight vector $b \in \{0, 1\}^n$. This conjectured form implies that a heavy Fourier coefficient can be found within quasipolynomial time, and leveraged to obtain nontrivial advantage in distinguishing the function from random.

We demonstrate that both pieces of evidence supporting the (more conservative) conjecture of Akavia et al. [1] apply as well to our strengthened variant. Namely, the conjecture provably holds for the case of:

- Arbitrary $g_k \in \text{AC0}$ and “typical” public matrices G , including random matrices with high probability. More concretely, any G for which $G \cdot x$ for uniform inputs x fools AC0 .
- Arbitrary public G , and g_k of polynomial size and depth 2 (i.e., CNF/DNF).

We observe that Akavia et al.’s proof for the former immediately applies to our setting as well; namely, the heavy Fourier coefficient they demonstrate already is of the desired form. The latter claim holds via a more subtle extension of the argument of Jackson [36], beyond the treatment within [1].

Relation between conjectures. We map the relation between the various conjectures posed within this work and beyond (depicted in Sect. 4.3, Fig. 1).

In particular, we draw a connection between our results and the linear IPPP conjecture of Servedio and Viola [56]: we observe that the nonexistence of WPRF in AC0 over public parities (which follows from our conjecture above), together with the *existence* of a WPRF in $\text{AC0} \circ \text{MOD}2$ (for which we provide a candidate) implies the Linear IPPP conjecture.

A related but technically incomparable observation was recently made in [27], which proves under a standard cryptographic assumption (namely, the learning with rounding assumption [9]) that either (1) the known quasipolynomial time learning algorithm for AC0 under the uniform distribution [41] cannot be extended to all \mathbb{F}_2 -linear distributions, even with subexponential time, or (2) an IPPP-style hardness conjecture is true, in the sense that $\text{AC0} \circ \text{MOD}2$ cannot compute inner-products *over the integers* (as opposed to inner product modulo 2). The paper also achieves related results under the assumption underlying the WPRF candidate of [14]. Our result is incomparable: it relies on new assumptions regarding the security of WPRF candidates in $\text{AC0} \circ \text{MOD}2$ instead of standard cryptographic assumptions, but applies to the “true” Linear IPPP conjecture instead of a variant over the integers.

Between Lapland and Cryptomania. Finally, we put forth the study of a new family of LPN-style assumptions, called *LPN with simple deterministic noise*. Roughly, these assumptions assert that one cannot distinguish pairs $(x, \langle x, s \rangle \oplus g_k(x))$ with random x from random pairs (x, y) , where s is a secret vector, and g_k is a *simple* secret function sampled at random from a family. By simple, we mean that g_k should belong to a low complexity class (such as $\text{AC0}[\text{MOD}2]$).

To our knowledge, this flavor of the learning parity with noise problem has never been studied; it bears some resemblance but is incomparable to the *learning parity with structured noise* framework of Arora and Ge [7], which consider noise patterns which are not deterministic, but satisfy some structure (typically, being roots of a low degree polynomial). This LPN with simple noise formulation captures the candidate weak PRF of [1], our candidate weak PRF in $\text{AC0} \circ \text{MOD}2$, and a recent candidate WPRF from [14] that can be viewed as being based on Learning with Rounding (LWR) [9] modulo 6. In the full version of this paper we

formulate a list of simple combinatorial properties of the noise function that we conjecture to be sufficient for the resulting candidate to defeat all *linear attacks*.

Further, we show that any candidate weak PRF in $\text{AC0}[\text{MOD2}]$ *implies* the existence of a hard instance of learning parity with simple noise. Weak PRFs in $\text{AC0}[\text{MOD2}]$ therefore necessarily live in “**Lapland**”, where there exist some codes (and deterministic noise distributions) for which the learning parity with noise assumption is hard. [1] describe a natural conjecture which implies that LPN is necessary for WPRFs in $\text{AC0} \circ \text{MOD2}$. Our result strengthens this, since it is unconditional and applies to the whole of $\text{AC0}[\text{MOD2}]$; on the other hand, we only show hardness of a specific instance of learning parity with simple noise, rather than standard LPN. It is an interesting open question to obtain a more natural LPN implication from candidate weak PRFs in $\text{AC0}[\text{MOD2}]$.

Our approach uses a result of Razborov and Smolensky [53, 57], who show that any $\text{AC0}[\text{MOD2}]$ function can be approximated by a low-degree polynomial; we show that the approximation noise itself can be used to define a learning parity with noise instance that fits our framework. On the other hand, we observe that the Razborov-Smolensky approximation could also be leveraged in a positive sense, for improving efficiency when evaluating the PRF homomorphically on ciphertexts or as part of a secure computation. For more details we refer to [10].

Note that the seeming contradiction of the Razborov-Smolensky approximation being sufficiently noisy to avoid decoding attacks (as far as we know), but precise enough to be useful for replacing the weak PRF by its approximation in applications, can be explained by the different number of input-output pairs considered in both contexts. An attacker attempting to break the security requires at least a quasipolynomial number of samples (because the low-degree multivariate polynomial potentially consists of a quasipolynomial number of terms), thus noise will occur *almost certainly*, whereas in an honest setting, when only computing a polynomial number of samples, likely the approximation will be perfect on all samples considered.

Since Lapland only has partial overlap with Cryptomania (that is, presently only LPN with low noise rate is known to imply public-key encryption with more than quasi-polynomial security³), one can further ask where in the regime between Lapland and Cryptomania weak PRFs in $\text{AC0}[\text{MOD2}]$ fall.

We put forward a second framework for “variable-density learning parity with noise (VDLPN) assumptions” into which the recent candidate weak PRF of [15] (who coined the term variable-density learning parity with noise for a specific instance of this broader framework) and our weak PRF candidate computed by sparse polynomials fall into. We further observe that any weak PRF candidate within this framework implies an instance of learning parity with simple deterministic noise with noise rate below the bound of [3]. This still does not imply public-key encryption, because the framework of [3] requires the code distribution to be dense, which is not the case for variable density learning parity with

³ More precisely, by a result of [3], random LPN implies public-key encryption if the noise rate is in $o(\sqrt{M})$, where M is the length of the secret.

noise. We still view this as an indication that weak PRFs within this framework “morally” live in Cryptomania.

To formalize this intuition, we put forward a conjecture, stating that with respect to some fixed noise rate, either *all codes* are efficiently decodable, or *almost all codes* are hard to decode. This is backed-up by the common understanding that LPN is in fact hard for random codes when choosing reasonably dense noise. Based on this conjecture we can indeed prove that candidate weak PRFs within the VDLPN framework imply public-key encryption following the strategy of [3]. We are not aware of any such implication for general functions in $\text{AC0}[\text{MOD2}]$ such as our candidate weak PRF in $\text{AC0} \circ \text{MOD2}$, even if willing to assume this conjecture, because the flavor of learning parity with noise implied by Razborov-Smolensky does not give low enough noise rate.

This is particularly interesting in light of recent developments on constructing *pseudorandom correlation functions* [15], since candidate constructions of expressive correlations so far all rely on either the VDLPN assumption [15], factoring-based assumptions [51], or extremely low-noise LPN [23], which with our result in mind, all imply public-key encryption.

2 Preliminaries

We start by recalling some basic properties of Boolean functions. We mostly follow standard notations and terminology (see, e.g., [41, 50]), except that we identify parity functions with vectors in $\{0, 1\}^n$ instead of subsets $S \subseteq \{1, \dots, n\}$.

Boolean functions. A Boolean function is a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$. When considering the Fourier coefficients of a function f we will consider it as a function $f: \{0, 1\}^n \rightarrow \{1, -1\}$ by identifying an output $b \in \{0, 1\}$ with $(-1)^b$.

The set of all *real-valued* functions on the cube $\{0, 1\}^n$ is a 2^n -dimensional real vector space with an inner product defined by $\langle g, f \rangle = 2^{-n} \cdot \sum_{x \in \{0, 1\}^n} f(x) \cdot g(x)$. The *norm* of f is defined as $\|f\| = \sqrt{\langle f, f \rangle}$.

Definition 1 (Characters). For $y \in \{0, 1\}^n$, the character χ_y is defined as $\chi_y(x) = (-1)^{\langle x, y \rangle}$.

Note that $\{\chi_y\}_{y \in \{0, 1\}^n}$ forms an orthonormal basis of the space of all real-valued functions on $\{0, 1\}^n$. Further, for all $y, z \in \{0, 1\}^n$ it holds that $\chi_y \chi_z = \chi_{y \oplus z}$.

Definition 2 (Fourier coefficients). As $\{\chi_y\}_{y \in \{0, 1\}^n}$ forms a basis, we can write every real-valued function f on the cube as $f = \sum_{y \in \{0, 1\}^n} \hat{f}(y) \cdot \chi_y$, for real-valued coefficients $\hat{f}(y)$, called Fourier coefficients.

Note that this is well-defined, as $\{\chi_y\}$ forms a basis for all functions $f: \{0, 1\} \rightarrow \mathbb{R}$. Further, as $\{\chi_y\}$ forms an orthonormal basis, the Fourier coefficient corresponding to $y \in \{0, 1\}^n$ can be computed as $\hat{f}(y) = \langle f, \chi_y \rangle$. For every Boolean function f we have $1 = \|f\|^2 = \langle f, f \rangle = \sum_{y \in \{0, 1\}^n} \hat{f}(y)^2$.

Definition 3 (Degree). *The degree $\text{deg}(f)$ of a Boolean function f is defined as the maximal Hamming weight of a vector $y \in \{0, 1\}^n$ for which $\hat{f}(y) \neq 0$.*

It can be shown that the above notion of degree coincides with standard algebraic degree.

Circuit classes. The class AC0 is the class of functions computed by a family of constant-depth, polynomial-size circuits of over AND/OR gates of unbounded fan-in along with negations. The class $\text{AC0} \circ \text{MOD2}$ is defined similarly, except that one also allows parity (XOR) gates *only at the bottom*. This can be viewed as applying an AC0 function to an \mathbb{F}_2 -linear encoding of the input. We define the circuit *depth* to be the length of the longest path from an input to an output, not counting negations. For instance, a DNF formula has depth 2. For $\text{AC0} \circ \text{MOD2}$ circuits we will consider by default only the depth of the AC0 part, namely ignoring parities. See, e.g., [1, 19, 56] for known facts about AC0 and $\text{AC0} \circ \text{MOD2}$.

In the context of cryptographic primitives, we will consider AC0 or $\text{AC0} \circ \text{MOD2}$ circuit families $\{C_\lambda\}$, parameterized by a *security parameter* λ , where the input length $n = n(\lambda)$ is assumed to be a monotonically-increasing, polynomially-bounded function of λ . We assume by default that such a circuit family is *polynomial-time uniform*, namely there is a polynomial-time algorithm whose output on input 1^λ is a description of C_λ ; however, we drop the uniformity requirement in the context of negative results.

2.1 Pseudorandom Functions

We consider here *weak* PRFs, which relax standard PRFs by only considering distinguishers that get the outputs of the function on uniformly random inputs. We require *subexponential* security by default, namely security against distinguishers of size 2^{n^ϵ} for some $\epsilon > 0$. This is the typical level of security achieved by constructions based on the strongest plausible versions of standard cryptographic assumptions. We formally define this notion below.

Definition 4 ((Weak) pseudorandom function [31, 46]). *Let $\lambda \in \mathbb{N}$ denote a security parameter and $n = n(\lambda)$, $\kappa = \kappa(\lambda)$ be monotonically-increasing and polynomially-bounded input length and key length functions, respectively.*

A (weak) pseudorandom function is syntactically defined by a function family $\mathcal{F} = \{f_\lambda: \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}\}$, where the output $f_\lambda(k, x)$ can be computed from (k, x) in polynomial time. Since λ and κ are determined by the input length n , we will sometimes write $f_k(x)$ instead of $f_\lambda(k, x)$.

For $T = T(\kappa)$ and $\epsilon = \epsilon(\kappa)$, we say that \mathcal{F} is a (T, ϵ) -secure strong pseudorandom function (PRF), if for every $\lambda \in \mathbb{N}$ and every oracle circuit \mathcal{A} of size $T(\kappa)$, it holds

$$\Pr_k[\mathcal{A}^{f_k(\cdot)} = 1] - \Pr_R[\mathcal{A}^{R(\cdot)} = 1] \leq \epsilon(\kappa),$$

where $\kappa = \kappa(\lambda)$, $k \xleftarrow{\$} \{0, 1\}^\kappa$ is chosen at random, and $R: \{0, 1\}^n \rightarrow \{0, 1\}$ is a truly random function. A T -secure PRF is a $(T, 1/T)$ -secure PRF.

We say that \mathcal{F} is a (T, ε) -secure weak PRF (WPRF) or T -secure WPRF if the above holds when \mathcal{A} only gets access to samples $(x_i, f_k(x_i))$, where $x_i \xleftarrow{\$} \{0, 1\}^n$ are chosen uniformly and independently. We say that \mathcal{F} is a (Q, T, ε) -secure (strong/ weak) PRF if \mathcal{A} only gets access to at most Q (chosen/ random) samples. Finally, we say that a (W)PRF \mathcal{F} has polynomial security if it is T -secure for every polynomial T , and that it has subexponential (resp., quasipolynomial, exponential) security if there exists $c > 0$ such that it is T -secure for $T = 2^{\kappa^c}$ (resp., $T = \kappa^{\log^c \kappa}$, $T = 2^{\kappa^c}$).

Our choice of subexponential security as the default level of security is motivated both from a cryptographic perspective and from an algorithmic perspective. From a cryptographic perspective, candidate PRFs with quasipolynomial security are relatively easy to obtain even in very low complexity classes and are considered “borderline insecure.” Subexponential (rather than exponential) security is typically the best level of security one can get from standard assumptions. From an algorithmic perspective, quasipolynomial-time algorithms (such as the LMN learning algorithm [41]) are considered “borderline efficient” and hence ruling out such algorithms requires PRFs with better than quasipolynomial security.

Finally, when referring to a (W)PRF \mathcal{F} in a circuit complexity class such as AC0 or $\text{AC0} \circ \text{MOD2}$, the default convention is that for each key sequence $k(\lambda)$, the induced function family f_k is in the class. We note that even when considered as a function of both the input and the key, our candidate constructions remain in $\text{AC0}[\text{MOD2}]$. On the other hand, our negative results and conjectures are stronger in that they apply to the fixed-key case and do not assume polynomial-time uniformity.

2.2 Preliminaries on Probability

Given t distributions $(\mathcal{D}_1, \dots, \mathcal{D}_t)$ over \mathbb{F}_2^n , we denote by $\bigoplus_{i \leq t} \mathcal{D}_i$ the distribution obtained by *independently* sampling $\mathbf{v}_i \xleftarrow{\$} \mathcal{D}_i$ for $i = 1$ to t and outputting $\mathbf{v} \leftarrow \mathbf{v}_1 \oplus \dots \oplus \mathbf{v}_t$.

Definition 5 (Bias of a Distribution). *Given a distribution \mathcal{D} over \mathbb{F}_2^n and a vector $\mathbf{u} \in \mathbb{F}_2^n$, the bias of \mathcal{D} with respect to \mathbf{u} , denoted $\text{bias}_{\mathbf{u}}(\mathcal{D})$, is equal to $\text{bias}_{\mathbf{u}}(\mathcal{D}) = \left| \frac{1}{2} - \Pr_{\mathbf{v} \xleftarrow{\$} \mathcal{D}} [\mathbf{u}^\top \cdot \mathbf{v} = 1] \right|$. Then, the bias of \mathcal{D} , denoted $\text{bias}(\mathcal{D})$, is defined as $\text{bias}(\mathcal{D}) = \max_{\mathbf{u} \neq \mathbf{0}^n} \text{bias}_{\mathbf{u}}(\mathcal{D})$.*

2.3 Algebraic Attacks and Rational Degree

Algebraic attacks have been introduced in [52] and were extended and abstracted in [20–22]. In its most basic form, an algebraic attack proceeds as follows: given a function $F : \{0, 1\}^n \mapsto \{0, 1\}$, it finds low degree multivariate polynomials (g, h) such that $F \cdot g = h$. If polynomials (g, h) of degree at most d are found, then the function F can be inverted given $n^{\tilde{O}(d)}$ random samples $(x, F(x))$. The

hardness of inverting a function with an algebraic attack is measured by its rational degree:

Definition 6 (Rational Degree). *The rational degree of a boolean function F is defined as the following quantity:*

$$\text{RD}(F) = \min_{g \neq 0} \{\text{deg}(g) \mid Fg = 0 \vee (F \oplus 1)g = 0\}.$$

Observe that the smallest d such that there exist polynomials (g, h) of degree at most d satisfying $F \cdot g = h$ necessarily satisfies $d \geq \text{RD}(F)$.

3 WPRFs by Sparse Multivariate Polynomials

In this section, we put forth a new candidate WPRF in a very low subclass of $\text{ACO}[\text{MOD}2]$: the class of *sparse multivariate polynomials* over \mathbb{F}_2 . That is, the key defines a sum of $\text{poly}(n)$ monomials in the inputs x_1, \dots, x_n . We conjecture that our candidate achieves subexponential security. To our knowledge, this is the first proposal for a WPRF in this class with plausible subexponential security.

In more detail, our candidate is inspired by a WPRF candidate from [15], which belongs to the class of *XNF formulas*, i.e., sparse polynomials in the inputs *and their negations*. Multivariate polynomials are an important object of study in learning theory. Our candidate WPRF provides an explicit distribution \mathcal{D} over sparse n -variate \mathbb{F}_2 -polynomials such that the following plausibly holds: there is a constant $\varepsilon > 0$ such that no 2^{n^ε} -time algorithm, given the values of a polynomial p sampled from \mathcal{D} on uniformly random inputs, can predict the value of p on a fresh random input with better than 2^{-n^ε} advantage. In contrast, the candidate of [15] only implies hardness of learning sparse polynomials under a somewhat artificial input distribution: the distribution over vector pairs (\mathbf{x}, \mathbf{y}) where \mathbf{y} is the bitwise negation of \mathbf{x} . To our knowledge, the only previous results in this setting are limited to showing *quasi-polynomial* hardness of learning sparse \mathbb{F}_2 -polynomials under the uniform distribution [24]. Our candidate complements the results of [34], which imply a $2^{\tilde{O}(\sqrt{n})}$ -time learning algorithms for sparse \mathbb{F}_2 -polynomials.

To support the conjectured subexponential security of our new candidate, we first observe that known results imply that it cannot be broken by *algebraic attacks*, as defined in Sect. 2. Furthermore, we show that its security can be formulated as an *LPN-style* assumption, which closely resembles (but is technically incomparable to) the variable-density learning parity with noise assumption of [15]. We provide support for the security of the candidate by proving that it cannot be broken in subexponential time by any *linear attack*, a large class of attacks which captures essentially all known attacks against LPN and its variants. Our analysis builds upon, but does not follow from, the analysis of [15]. In the full version we elaborate on the specific challenges that arise when trying to extend the analysis of [15] to our candidate.

3.1 Our Candidate

Our candidate builds upon the candidate of [15], which was carefully crafted as a XOR of variable-size terms (products of variables and negated variables), where the purpose of terms of size i is to defeat all linear attacks that depend on (approximately) 2^i samples. In [15], the set of input variables in each term is fixed in advance; the WPRF key simply tells, for each variable in each term, whether to use the input or its negation. To confine our candidate to the subclass of sparse \mathbb{F}_2 -polynomials, we must refrain from using negations of inputs. This suggests a very natural variant: instead of selecting between bits x and $1 - x$, the key is used to randomly select one out of b random bits $x_1 \cdots x_b$ for each variable of each monomial. When b is large enough, since the fraction of zeroes and ones in random b -bit strings is tightly concentrated around $1/2$, this intuitively provides security guarantees comparable to that of [15]. We formally introduce the candidate below.

- **Input domain:** $x \in \{0, 1\}^n$ with $n = w \cdot D \cdot (D - 1) \cdot b/2$. We view x as a concatenation of D blocks $(x_i)_{i \leq D}$, where block x_i contains w sub-blocks $x_{i,1}, \dots, x_{i,w}$, and each sub-block $x_{i,j}$ is composed of i b -bit strings $(x_{i,j,\ell})_{\ell \leq i}$. Given a string $x_{i,j,\ell}$, we write $x_{i,j,\ell}[k]$ to denote its k -th bit.
- **Key domain:** $K = (K_{i,j,\ell})_{i \leq D, j \leq w, \ell \leq i} \in \{b\}^s$ with $s = w \cdot \sum_{i=1}^D i$.
- **Candidate:**

$$F_K(x) = \bigoplus_{i=1}^D \bigoplus_{j=1}^w \bigwedge_{\ell=1}^i x_{i,j,\ell}[K_{i,j,\ell}]$$

Security against algebraic attacks. The security of our candidate against algebraic attacks [22] follows directly from a known bound on the rational degree of triangular functions.

Lemma 7. *For any $K \in \{b\}^s$, an algebraic attack in the sense of [22] requires (time and) number of samples lower bounded by $n^{\Omega(D)} = 2^{\Omega(D \log(D+w+b))}$.*

Lemma 7 follows readily from the fact that our candidate weak PRF has high *rational degree*: for any $K \in \{0, 1\}^s$, it holds that $\text{RD}(F_K) \geq D$. The proof follows immediately from [42]: for any fixed choice of key K , F_K is a direct sum of w independent *triangular functions of degree D* , each evaluated on distinct portions of the input, where (denoting $D' = D(D - 1)/2$) the triangular function of degree D is the function $T_D(x_1, \dots, x_{D'}) = x_1 \oplus x_2 x_3 \oplus \dots \oplus \bigwedge_{\ell=D'-D}^{D'} x_\ell$. By Lemma 3 of [42], the rational degree of a direct sum of functions is at least the largest rational degree of its components, and by Lemma 6 of [42], the rational degree of T_D is exactly D .

3.2 Variable-Density LPN Formulation

We now show that the security of our weak PRF candidate follows from a VDLPN-style assumption, in the spirit of [15]. We note, however, that the concrete assumption is not directly comparable to that of [15]: while the corresponding noise distributions are similar, the variable-density matrix distribution

in our work is very different. In the following, for each $(i, j) \in [D] \times [w]$, it is convenient to view $K_{i,j} = (K_{i,j,\ell})_{\ell \leq i}$ as a single integer from the set $[b^i]$, via the natural embedding. Then, let $\mathbf{u}(K_{i,j})$ denote the unit length- b^i vector with a 1 at position $K_{i,j}$ and 0's elsewhere. We can rewrite F_K as

$$\begin{aligned}
 F_K(x) &= \bigoplus_{i=1}^D \bigoplus_{j=1}^w \left\langle \bigotimes_{\ell=1}^i x_{i,j,\ell}, \mathbf{u}(K_{i,j}) \right\rangle \\
 &= \left\langle x_{1,1,1} \|\cdots\| \bigotimes_{\ell=1}^D x_{D,w,\ell}, \mathbf{u}(K_{1,1}) \|\cdots\| \mathbf{u}(K_{D,w}) \right\rangle = \langle h(x), e(K) \rangle
 \end{aligned}$$

where $h : x \rightarrow (x_{1,1,1} \|\cdots\| \bigotimes_{\ell=1}^D x_{D,w,\ell})$ and $e : K \rightarrow (u(K_{1,1}) \|\cdots\| \mathbf{u}(K_{D,w}))$. Now, given a bound N on the number of samples, we let $\mathcal{H} = \mathcal{H}(D, w, \mathbf{b}, N)$ denote the distribution over matrices H in $\mathbb{F}_2^{N \times (w \cdot \sum_{i=1}^D b^i)}$ whose N rows are sampled as $h(x)$ for independent samples $x \xleftarrow{\$} \{0, 1\}^n$. Furthermore, we let $\mathcal{N} = \mathcal{N}(D, w, \mathbf{b})$ denote the distribution over vectors \mathbf{e} in $\mathbb{F}_2^{w \cdot \sum_{i=1}^D b^i}$ induced by sampling $K \xleftarrow{\$} [b]^s$ and outputting $e(K)$. Clearly, breaking the security of our candidate given N samples is equivalent to breaking the $(\mathcal{H}, \mathcal{N})$ -dualLPN assumption. This variant of the dual LPN assumption is very close in spirit to the regular VDLPN assumption from [15]: the noise distribution is the same up to setting $\mathbf{b} = 2$. The matrix distribution, on the other hand, is quite different, but satisfies the same sparsity condition: the matrix H is divided into D submatrices H_i , and the average sparsity of the rows of H_i is $(w \cdot (b/2)^i) / (w \cdot b^i) = 1/2^i$. The matrix distribution in [15] satisfies the same variable density structure, which motivated the name ‘‘variable-density LPN’’. Therefore, we view our new candidate as belonging to the same family of LPN variants.

3.3 Security Against Linear Attacks

We turn to consider the class of *linear attacks*, which in the context of pseudorandom *generators* captures the notion of small-bias generators [45]. Linear attacks capture, intuitively, every attack where the distinguisher is restricted to compute a linear function of the LPN samples, the identity of which can be arbitrarily determined from the public LPN matrix and inputs. This captures essentially all known attacks against standard variants of LPN, such as those based on Gaussian elimination, statistical decoding, information set decoding, and BKW-style attacks. The work of [15] provided support for their VDLPN conjecture by proving subexponential security against such linear attacks.

In the context of a WPRF, a linear distinguisher is first given N random inputs x_1, \dots, x_N , and then must choose a subset of indices $S \subset \{1, \dots, N\}$ such that the distribution $\bigoplus_{i \in S} f_k(x_i)$, for a random choice of k , is biased towards 0 or 1. More formally, we use the following notion of an (ϵ, δ, N) -biased WPRF, which naturally extends the standard notion of an ϵ -biased pseudorandom generator.

Definition 8 ((ε, δ, N) -biased weak PRF family, [15]). A function family $\{F_K : \mathbb{F}_2^{n(\lambda)} \mapsto \mathbb{F}_2\}_{K \in \mathbb{F}_2^{s(\lambda)}}$ is (ε, δ, N) -biased if for every large enough $\lambda \in \mathbb{N}$, letting $\mathcal{D}_{\lambda, N}(\mathbf{x})$ (for some $\mathbf{x} \in (\mathbb{F}_2^{n(\lambda)})^N$) biased with inputs of length samples $K \xleftarrow{\$} \mathbb{F}_2^{s(\lambda)}$ and outputs $\mathbf{y} = (F_K(x^{(1)}), \dots, F_K(x^{(N)}))$, it holds that

$$\Pr_{x^{(1)}, \dots, x^{(N(\lambda))} \xrightarrow{\$} \mathbb{F}_2^{n(\lambda)}} [\text{bias}(\mathcal{D}_{\lambda, N}(\mathbf{x})) > \varepsilon(\lambda)] \leq \delta(\lambda).$$

Notation and theorem statement. We first introduce some notation. Recall that a sample H from \mathcal{H} is a concatenation of D matrices H_i , where each matrix H_i is itself a concatenation of w submatrices $H_{i,j} \in \mathbb{F}_2^{N \times b^i}$ whose rows are of the form $\bigotimes_{\ell=1}^i x_{i,j,\ell}$, where the $(x_{i,j,\ell})_{\ell \leq i}$ are i uniformly random independent b -bit strings. For any fixed matrix H in the support of \mathcal{H} , we let $\mathcal{D}_{\text{out}}(H)$ denote the distribution induced by sampling $\mathbf{e} \leftarrow \mathcal{N}$ and outputting $H \cdot \mathbf{e}$.

Theorem 9 (Low bias). Fix a security parameter λ . There exist constants $0 < \beta, \nu, \mu < 1$ such that for any parameters (D, w, \mathbf{b}, N) satisfying $w = \text{poly}(\lambda)$, $\mathbf{b} = \text{poly}(\lambda)$, $D^2 \leq \beta \cdot w$, $D \leq \frac{\sqrt{b}}{2\lambda} + 1$, and $N \leq 2^D$, letting $\mathcal{H} = \mathcal{H}(D, w, \mathbf{b}, N)$, it holds that

$$\Pr_{H \leftarrow \mathcal{H}} [\text{bias}(\mathcal{D}_{\text{out}}(H)) > \mu^w] \leq \nu^D + \nu^{\lambda^2}.$$

For example, using the choice of parameters $(D, w, \mathbf{b}, N) = (\lambda, \lambda^2/\beta, 4\lambda^4, 2^\lambda)$, our candidate is $(2^{-\Omega(\lambda^2)}, 2^{-\Omega(\lambda)}, 2^\lambda)$ -biased with inputs of length $O(\lambda^8)$, and keys of length $\tilde{O}(\lambda^4)$.

To facilitate comparison with the analysis of [15], we let \mathcal{H}' and \mathcal{N}' denote respectively the matrix and noise distributions for the VDLPN variant of [15], where a sample $H \leftarrow \mathcal{H}'$ can also be broken into D matrices $H_i = H_{i,1} \parallel \dots \parallel H_{i,w}$ where the $H_{i,j}$ are independent matrices; we denote by \mathcal{H}'_i the distribution over H_i induced by $H \leftarrow \mathcal{H}'$ for any $i \leq D$.

High level overview. At a high level, the security analysis follows the same approach as the analysis in [15] (which should come as no surprise due to the similarities between the candidates); however, the analysis is significantly more involved due to the more complex structure of the matrix distribution for our candidate. Fix $i \leq D$. The analysis of [15] proceeds roughly as follows.

1. Using a strong concentration bound (McDiarmid’s bounded difference inequality), it shows that for any fixed *attack vector* $\mathbf{v} \in \mathbb{F}_2^N$ whose Hamming weight is between 2^{i-1} and 2^i , except with probability at most $\exp(-\Omega(w \cdot 2^i))$, a random matrix $H_i \leftarrow \mathcal{H}'_i$ satisfies $\mathcal{HW}(\mathbf{v}^\top \cdot H_{i,j}) / |\mathbf{v}^\top \cdot H_{i,j}| \in [\varepsilon, 1 - \varepsilon]$, where ε is some constant (that is, $\mathbf{v}^\top \cdot H_{i,j}$ has a fraction of ones bounded by a constant, and bounded away from 1 by a constant), for a fraction at least $w/2$ of the w submatrices $H_{i,j}$ of H_j . Such a matrix H_i is called *good with respect to* \mathbf{v} .

2. From a union bound over all vectors \mathbf{v} of weight between 2^{i-1} and 2^i , it follows that, except with probability at most $\exp(-\Omega((\log N - w) \cdot 2^i))$, a random matrix $H_i \leftarrow \mathcal{H}'_i$ will be good with respect to all vectors \mathbf{v} in this weight range. When w is sufficiently larger than $\log N$, this probability is bounded by $\exp(-\Omega(w))$ for any $i \leq D$.
3. By a union bound over all $i \leq D$, with probability at least $1 - D \cdot \exp(-\Omega(w)) = 1 - \exp(-\Omega(w))$, a random matrix $H \leftarrow \mathcal{H}'$ satisfies the following: for every nonzero vector \mathbf{v} , there is an $i^* \leq D$ such that H_{i^*} is good with respect to \mathbf{v} . Then, for any such matrix H , $H \cdot \mathbf{e}$ for $\mathbf{e} \leftarrow \mathcal{N}'$ is the vector obtained by sampling a uniformly random column from each $(H_{i,j})_{i \leq D, j \leq w}$ and XORing them all. Since H_{i^*} is good with respect to \mathbf{v} , $H \cdot \mathbf{e}$ will include at least $w/2$ terms sampled randomly and independently from bitstrings $\mathbf{v} \cdot H_{i^*,j}$ with a fraction of ones in $[\varepsilon, 1 - \varepsilon]$. It follows that, with probability at least $1 - \exp(-\Omega(w))$ over the random choice of $H \leftarrow \mathcal{H}'$, the distribution of $H \cdot \mathbf{e}$ for $\mathbf{e} \leftarrow \mathcal{N}'$ has bias with respect to \mathbf{v} at most $(1 - \varepsilon)^{w/2}/2 = 2^{-\Omega(w)}$, for any possible nonzero vector \mathbf{v} .

Looking ahead, our security analysis will follow the same three steps as above, and the steps 2 and 3 will be the same as in [15]. However, while the first step also consists in proving a similar bound, the actual analysis turns out to be much more involved due to the different matrix structure. Due to space limitations, the proof of Theorem 9 is deferred to the full version.

4 WPRFs in $\text{AC0} \circ \text{MOD2}$

In this section we present a candidate construction of a weak PRF in $\text{AC0} \circ \text{MOD2}$ (recall, unlike $\text{AC0}[\text{MOD2}]$, here the parity gates must lie at the input layer of the circuit). We follow the high-level template of Akavia et al. [1]. Their construction, referred to as ABGKR, is of the form

$$f_{s,K}(x) = \langle x, s \rangle \oplus g(K \cdot x \pmod 2)$$

for $s \in \{0, 1\}^n$, $K \in \{0, 1\}^{(n-1) \times n}$, where $g(x) = \bigvee_{i=1}^{\lambda} \bigwedge_{j=1}^{\log \lambda} x_{ij}$ is a DNF (the so-called TRIBES function). Since $f_{s,K}(x)$ can be written as $(\neg \langle x, s \rangle \wedge g(K \cdot x)) \vee (\langle x, s \rangle \wedge \neg g(K \cdot x))$, it indeed belongs to $\text{AC0} \circ \text{MOD2}$.

The rationale behind the design of Akavia et al. is the following: even when picking a very simple function g (in their case, a DNF), the function $g_K(x) = g(K \cdot x)$ can already not be distinguished from a random c -unbalanced function (i.e. a random function f with $\Pr_x[f(x) = 1] = c$ for some constant $c \neq 1/2$) for various natural attacks (e.g. correlations with small function families and closeness to low-degree polynomial). Then, this function g_K is XORed with $\langle x, s \rangle$ to make the final function balanced.

From unbalanced WPRFs to standard WPRFs. We observe that this transformation does actually provably turn an unbalanced WPRF into a “standard” WPRF, under the LPN assumption. The proof of this observation is

straightforward; for details we refer to the full version. In spite of its simplicity, this observation had to our knowledge never been made.

We further note that there exists an alternative, unconditional transformation from a c -unbalanced WPRF in $\text{AC0} \circ \text{MOD2}$ into a standard WPRF in $\text{AC0} \circ \text{MOD2}$ which relies on the Von Neumann randomness extractor: assume w.l.o.g. that $c < 1/2$. Use (say) $2n$ parallel instances of the c -unbalanced WPRF on independent inputs and keys, grouped into n pairs. Then, take the first pair of distinct output bits (since c is a constant, there is one such pair with overwhelming probability $1 - 2^{-O(n)}$): if it is 01, define the output of the WPRF to be 0; else, define it to be 1. It is relatively straightforward to prove that if g_k is a c -unbalanced WPRF, the resulting function is a WPRF. This process can be executed in AC0 , hence the resulting function is in $\text{AC0} \circ \text{MOD2}$.

Our approach. The above discussion justifies focusing on the task of building *unbalanced* WPRFs in $\text{AC0} \circ \text{MOD2}$, since the latter imply standard WPRFs in the same class through simple transformations. The ABGKR candidate instantiates this unbalanced WPRF with a DNF on top of parities; however, the attack of [13] allows to distinguish *any* depth-2 AC0 circuit on top of parities from unbalanced random functions, since any such function must have low rational degree. Therefore, any unbalanced WPRF in $\text{AC0} \circ \text{MOD2}$ must have at least three layers of AND/OR gates. With the goal of finding the simplest possible modification of the ABGKR candidate which can retain subexponential security, we ask:

Is there a subexponentially secure unbalanced WPRF computable by a depth-3 AC0 circuit on top of parities?

Our candidate. We put forth the following candidate unbalanced WPRF: $g_k(x) = g(K \cdot x)$, with

$$g(x) = \bigvee_{i=1}^{\lambda} \bigwedge_{j=1}^{\lambda} \bigvee_{k=1}^w x_{ijk}, \tag{1}$$

where λ is a security parameter (i.e., we will bound the complexity of various attacks on our candidate as a function of λ) and m, w are chosen such that $w = \lceil \log \lambda - \log \log \lambda \rceil$ and $m = \lambda^2 w$. That is, we simply add a single layer of ORs after the parity layer, with parameters chosen to guarantee that $\Pr_x[g(x) = 1]$ is constant. Note that choosing the fan-in of the gates more carefully, one can actually obtain bias $1 = 2 + o_n(1)$. In this case the function $g(x)$, which replaces the TRIBES function in ABGKR, corresponds to the degree-3 Sipser function. For more details, we refer to [33, 55].

We conjecture that this candidate achieves subexponential security. Observe that since the attack of [13] distinguishes any depth-2 AC0 circuit on top of

parities from unbalanced random functions, our candidate actually enjoys optimal depth.⁴

4.1 Provable Resistance to Algebraic Attacks

Algebraic attacks are a general class of cryptanalytic algorithms that aim to either invert a function or distinguish it from random, by obtaining many samples and using these to derive a system of linear equations over the secret inputs. This class of attack was first developed by the applied cryptographic community and used to break public-key encryption schemes and stream ciphers [20, 22, 52]. It generalizes in particular the correlation attacks [37] that have been developed for attacking LFSRs. Correlation attacks have been considered in the theory community in the context of constructing local pseudorandom generators [44].

The resistance of a WPRF $f_k : \{0, 1\}^n \rightarrow \{0, 1\}$ to algebraic attacks can be measured by its *rational degree*, that is, the smallest d for which there exist non-zero polynomials p and q of algebraic degree at most d , such that

$$f_k(x) \cdot p(x) = q(x), \quad \forall x \in \{0, 1\}^n. \quad (2)$$

Applebaum and Lovett [5] formally studied algebraic attacks of local functions, and showed that if a predicate has large rational degree then it provably resists a natural class of algebraic attacks.

On the other hand, if a WPRF candidate f_k has low rational degree d , then it can be distinguished from random via a simple algebraic attack, which obtains $O(n^d)$ samples and tests whether (2) holds for each of them. This is exactly the type of attack that Bogdanov and Rosen [13] observed breaks the candidate of Akavia et al. [1] in quasipolynomial time, since it has rational degree $O(\log \lambda)$.

We, on the other hand, show that our candidate has rational degree λ . Even though, formally, this does not rule out the attack of [13], which only requires proximity to low rational degree, we view this as strong evidence that the attack does not apply to our candidate.

To analyze the rational degree of our candidate, we first give a general method for determining the exact rational degree of any function in AC_0 that can be expressed as alternating layers of AND and OR gates that each depend on disjoint subsets of the input. We then use this to compute the rational degree of our noise function, and finally our candidate unbiased WPRF.

Towards understanding our techniques, we first briefly recall the attack of Bogdanov and Rosen [13]. To that end, note that the rational degree can be characterized as the minimal d such that there exists a polynomial $p \neq 0$ of algebraic degree d such that $f \cdot p = 0$ or $(f \oplus 1) \cdot p = 0$ (also referred to as the *algebraic immunity* in the literature). The attack of Bogdanov and Rosen [13]

⁴ However, transforming our candidate into a standard WPRF, e.g. using the LPN-based transformation, results in a candidate computed by a depth-4 AC_0 circuit on top of parities. It is an interesting question whether the optimal depth can be achieved for standard WPRFs, i.e., whether there exists subexponentially-secure standard WPRFs computable by depth-3 AC_0 circuit on top of parities.

builds on the observation that $f = \bigvee f_i$ always has rational degree at most $\min_i \deg f_i$, as $f_i(x) = 1$ implies $f(x) = 1$ and thus $(f \oplus 1) \cdot f_i = 0$. Therefore, for any DNF either all inner conjunctions have high algebraic degree (and thus the DNF is highly biased towards 0), or the function is susceptible to rational degree attacks.

We observe that while a disjunction does not increase the rational degree of a function, it does have an effect that can be leveraged. Namely, consider a function $p \neq 0$ of minimal algebraic degree such that $f \cdot p = 0$. We will prove that if p_i are the minimal annihilating functions for f_i (and all functions depend on disjoint parts of the input), p must have algebraic degree at least $\sum_i p_i$.

Now, using that conjunctions behave in a dual way, alternating between conjunctions and disjunctions allows to increase the rational degree while keeping the function's bias constant. In order to prove this, we introduce the notion of *primal* and *dual* rational degree.

Definition 10 (Primal and dual rational degree). For $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we define the primal rational degree ρ as the minimal ρ such that there exists a polynomial $p \neq 0$ with algebraic degree ρ and $f \cdot p = 0$. Further, we define the dual rational degree ρ' of f as the primal rational degree of its negation. Namely, we define the dual rational degree as the minimal ρ' such that there exists a polynomial $p \neq 0$ with algebraic degree ρ' and $(f \oplus 1) \cdot p = 0$. Note that the rational degree of f is $d = \min(\rho, \rho')$.

With the notion of primal and dual rational degree we can distill our main observation in the following lemma, which we prove in the full version.

Lemma 11. Let $f, h: \{0, 1\}^n \rightarrow \{0, 1\}$ be Boolean functions that depend on disjoint parts of the input⁵, where f and h have primal rational degree ρ_f and ρ_h and dual rational degree ρ'_f and ρ'_h , respectively. Then:

- (i) The primal rational degree of $f \vee h$ is lower bounded by $\rho_f + \rho_h$.
- (ii) The dual rational degree ρ' of $f \vee h$ is lower bounded by $\min(\rho'_f, \rho'_h)$.

With this, it is straightforward to compute the exact rational degree of a disjunction, where all terms depend on disjoint parts of the input. Similarly, we can also apply this to compute the rational degree of a conjunction, since $\bigwedge_{i=1}^s f_i = \bigvee_{i=1}^s (f_i \oplus 1) \oplus 1$.

Put together, and applied to our candidate, we obtain the following.

Lemma 12. Let $m = m(\lambda) \in \mathbb{N}$, let $g: \{0, 1\}^m \rightarrow \{0, 1\}$ be as in Eq. 1, let $n = m + 1$, and let $s \in \{0, 1\}^n, K \in \{0, 1\}^{m \times n}$ be such that the map $x \mapsto (\langle x, s \rangle, K \cdot x) \pmod 2$ is invertible. Then, our candidate weak PRF $f_{s,K}: \{0, 1\}^n \rightarrow \{0, 1\}$ defined via

$$f(x) \mapsto \langle x, s \rangle + g(K \cdot x \pmod 2)$$

has rational degree at least λ .

For further details and discussion, we refer the reader to the full version.

⁵ We say that f depends on the i -th index of the input, if x_i appears with a non-zero coefficient in some term in f .

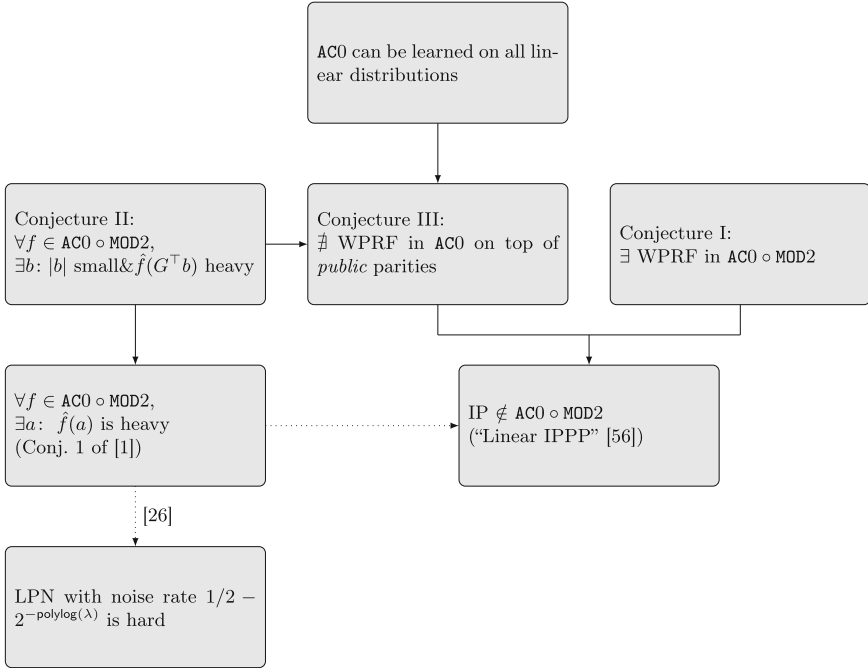


Fig. 1. Relation between different assumptions/ conjectures. $A \rightarrow B$ means that A implies B . By a linear distribution we mean the uniform distribution over a linear subspace $V \subseteq \{0, 1\}^n$, where dotted implications were already observed by [1].

4.2 On Resistance to Linear Attacks

We also consider the resistance of our candidate to linear attacks, as was done for our other candidate in Sect. 3. While we have not been able to prove resistance of linear attacks for this candidate, we formulate a combinatorial conjecture which states, informally, that if the deterministic noise function is c -unbalanced for some constant c and far from all low-degree polynomials, then no attack from the linear attack framework can break the corresponding LPN with simple noise assumption. If true, this conjecture would imply that our candidate, the ABGKR candidate, as well as the “LWR mod 6” candidate from [14], cannot be broken by any of the above attacks. We provide preliminary observations regarding the plausibility of the conjecture; we view proving or disproving this conjecture as an interesting open question. For more details we refer to the full version

4.3 On WPRFs in AC0 with Public Parities

In this work we give a candidate construction of a weak PRF in $AC0 \circ MOD2$, where the parities are *secret*. In particular, we conjecture that such a weak PRF exists (this is in the following referred to as Conjecture I).

We also consider the natural question of the existence of a simpler class of WPRF of the form $f_k(x) = g_k(G \cdot x)$, where G is a *public* matrix. Note that if G is removed (or surjective) then f_k could be learned by the algorithm of Linial, Mansour and Nisan [41] for learning AC0 under the uniform distribution.

While Akavia et al. [1] conjectured that any function in $\text{AC0} \circ \text{MOD2}$ has a large Fourier coefficient, we take this further by suggesting that, in the case of a public matrix G , the heavy Fourier coefficient of f_k stems from a low-order coefficient of g_k (in the following referred to as Conjecture II). This would imply that the high-weight Fourier coefficient can be used to distinguish the function from random in quasipolynomial time even given only access to random samples, and therefore allows to conclude that there cannot exist a weak PRF in AC0 on top of *public* parities (in the following referred to as Conjecture III).

We prove Conjecture II for the case when g_k is a family of DNFs, by extending the work of Jackson [36] to show that the coefficient is of the right form. The idea of Jackson is that any DNF correlates with a parity of its term that is “most likely” to be satisfied, which implies a heavy Fourier coefficient. We further observe that this means the function is either biased, or the term can contain only a few non-correlated variables. Since an AND clause is only satisfied for exactly one setting of inputs, if there are too many independent terms in the DNF then the function is biased. Otherwise, there are many dependencies between the individual terms, which we show implies the heavy Fourier coefficient comes from a vector of the form $a = G^\top v$ for some low-weight v .

We further prove Conjecture II for arbitrary $g_k \in \text{AC0}$ if the matrix G is random (or, more generally, defines a polylog-wise independent map).

We present the formal Conjectures I, II and III as well as the proof of Conjecture II for the above mentioned special cases in the full version.

Linear IPPP and Relations Between Conjectures. Finally, in the full version, we also elaborate on the relations between our conjectures, and previous conjectures in the literature including the “Linear IPPP” conjecture [56], asserting that mod-2 inner product is not in $\text{AC0} \circ \text{MOD2}$. These connections are illustrated in Fig. 1.

5 Between Lapland and Cryptomania

In this section we present two abstract frameworks. We first introduce the notion of learning parity with simple deterministic noise, which captures our candidate weak PRF in $\text{AC0} \circ \text{MOD2}$ from Sect. 4. Further, we show that every weak PRF candidate in $\text{AC0}[\text{MOD2}]$ implies some form of learning parity with simple deterministic noise.

Next, we introduce an abstract framework that captures variable-density learning parity with noise style assumptions such as the candidate weak PRF of [15] and our candidate weak PRF from Sect. 3.

Further, if one believes that either no code is hard to decode or almost all codes are hard to decode with respect to some noise level, then we show that each

candidate that fits into the VDLPN framework lives in Cryptomania. We are not aware of any similar implications for functions that can be cast as learning parity with $\text{AC0}[\text{MOD2}]$ -noise more generally.

5.1 Learning Parity with Simple Deterministic Noise

We observe that the Akavia et al. [1] candidate as well as our own candidate in $\text{AC0} \circ \text{MOD2}$ can be cast as a form of new LPN-style assumption, that we refer to as *LPN with simple deterministic noise*. This can be viewed as a generic method to transform a biased weak PRF into a weak PRF. Formally, we define learning parity with simple noise as follows.

Definition 13 (Learning parity with simple deterministic noise). *Let $n = n(\lambda), \kappa = \kappa(\lambda) \in \mathbb{N}$ and let $\mathcal{G} = \{g_k: \{0, 1\}^n \rightarrow \{0, 1\} \mid k \in \{0, 1\}^\kappa\}$ be a family of keyed functions in a low-complexity class. We say a function family $\mathcal{F} = \{f_{s,k}: \{0, 1\}^n \rightarrow \{0, 1\} \mid s \in \{0, 1\}^n, k \in \{0, 1\}^\kappa\}$ is an instance of learning parity with simple deterministic noise from \mathcal{G} , if $f_{s,k}: \{0, 1\}^n \rightarrow \{0, 1\}$ is of the form $f_{s,k}(x) = \langle x, s \rangle \oplus g_k(x)$.*

In this paper by simple we usually refer to noise functions in $\text{AC0}[\text{MOD2}]$. Note that if \mathcal{G} is in $\text{AC0}[\text{MOD2}]$, then so is \mathcal{F} . Further note that $f_{s,k}$ can be written as

$$f_{s,k}(x) = (\neg \langle x, s \rangle \wedge g_k(x)) \vee (\langle x, s \rangle \wedge \neg g_k(x)).$$

This shows that for $g_k \in \text{AC0} \circ \text{MOD2}$ we also have $f_{s,k} \in \text{AC0} \circ \text{MOD2}$ (where we consider the key as fixed). Note that this transformation from a biased weak PRF g_k to a weak PRF $f_{s,k}$ is not depth-preserving, however.

This framework can be extended to capture more general input distributions as follows.

Definition 14 (Extension to general input distributions). *Let $n = n(\lambda), \kappa = \kappa(\lambda), M = M(\lambda) \in \mathbb{N}$, let $\mathcal{G} = \{g_k: \{0, 1\}^n \rightarrow \{0, 1\} \mid k \in \{0, 1\}^\kappa\}$ be a family of keyed functions in a low-complexity class, and let $h: \{0, 1\}^n \rightarrow \{0, 1\}^M$ a function. We say that a function family $\mathcal{F} = \{f_{s,k}: \{0, 1\}^n \rightarrow \{0, 1\} \mid s \in \{0, 1\}^M, k \in \{0, 1\}^\kappa\}$ is an instance of learning parity with simple deterministic noise from \mathcal{G} with respect to the input distribution generated by h , if $f_{s,k}: \{0, 1\}^n \rightarrow \{0, 1\}$ is of the form $f_{s,k}(x) = \langle h(x), s \rangle \oplus g_k(x)$.*

Of course not every class of noise functions gives rise to a candidate weak PRF. In the full version we make progress on studying learning parity with simple noise by presenting a combinatorial conjecture about properties that the family of noise functions \mathcal{G} has to satisfy (informally speaking, these are the properties of being “balanced” and having “high-degree”), that we believe are sufficient in order to resist all linear attacks. However, note that as the attack by Bogdanov and Rosen [13] showed, satisfying these properties is still not sufficient to be a weak PRF, because other classes of attacks such as algebraic attacks might apply.

5.2 Weak PRFs in $\text{AC0}[\text{MOD2}]$ Live in Lapland

The results on circuit lower bounds by Razborov and Smolensky [53, 57] show that every function in $\text{AC0}[\text{MOD2}]$ can be approximated by a polynomial of poly-logarithmic degree. More formally, their result can be stated as follows.

Theorem 15 (Razborov-Smolensky [53, 57]). *Let $n, d, S \in \mathbb{N}$. If $f: \{0, 1\}^n \rightarrow \{0, 1\}$ can be computed by depth- d , size- S circuit with MOD2 gates, then for any integer $\varepsilon > 0$, there exists a polynomial $p(x) \in \mathbb{F}_2[x_1, \dots, x_n]$ of degree at most $(\log(S/\varepsilon))^d$ such that $\Pr_x[f(x) \neq p(x)] \leq \varepsilon$.*

The theorem implies that if there is a weak PRF in $\text{AC0}[\text{MOD2}]$, then learning parity with noise is hard, where the code is a “punctured” Reed-Muller code of quasipolynomial dimension (i.e. the row corresponding to an input x consists of the of all low-degree monomials evaluated on x), the secret corresponds to the coefficient of the polynomial that approximates the weak PRF, and the noise corresponds to the approximation error. In other words, the existence of a weak PRF in $\text{AC0}[\text{MOD2}]$ says that this kind of punctured Reed-Muller codes are hard to decode for some nontrivial noise rate.

Corollary 16. *Let $n = n(\lambda), \kappa = \kappa(\lambda) \in \mathbb{N}$. If there exists a (Q, T, ε) -weak PRF in $\text{AC0}[\text{MOD2}]$, then there exists $c, C \in \mathbb{N}$ with $c < C$, a family of keyed functions $\mathcal{G} = \{g_k: \{0, 1\}^n \rightarrow \{0, 1\} \mid k \in \{0, 1\}^\kappa\}$ and a function $h: \{0, 1\}^n \rightarrow \{0, 1\}^M$ where $M = 2^{\log^C \kappa}$, such that the learning parity function $f_{s,k} = \langle h(x), s \rangle \oplus g_k(x)$ with deterministic noise $g_k \in \mathcal{G}$ respective to the input distribution generated by h is a $(Q, \mathcal{O}(T), \varepsilon)$ -weak PRF. Further, for the corresponding noise rate we have that $\Pr_{x,k}[g_k(x) = 1] \leq 2^{-\log^c \kappa}$.*

Note that the Razborov-Smolensky result does not make any guarantees as to the *distribution* over the approximating low-degree polynomial for the functions in the PRF family, corresponding to distribution over the secret s in the LPN instance. However, the corresponding LPN instance reduces to the case of average-case s . Namely, samples $\langle x, s^* \rangle \oplus g_k(s)$ for arbitrary s^* can be generically converted to consistent samples for uniform secret $s^* + s'$, by offsetting each sample by $\langle x, s' \rangle$.

Note that having a superpolynomial secret in Corollary 16 only “scales down” the LPN security when expressed as a function of the secret size, and in the subexponential regime the resulting guarantee remains meaningful. More explicitly, the corollary can be understood as follows: If there exists a weak PRF in $\text{AC0}[\text{MOD2}]$ with subexponential security 2^{κ^δ} , then there exists an instance of deterministic LPN that has secret length $M = 2^{\log^C \kappa}$ and security in the order of $2^{\kappa^\delta} = 2^{2^{\delta \cdot \log^{1/C} M}}$. Thus, the existence of weak PRF candidates in $\text{AC0}[\text{MOD2}]$ with subexponential security implies what can be viewed as an instance of deterministic LPN with “subsubexponential hardness” in the secret length (which lies strictly between quasipolynomial and subexponential).

Consider a hardness of decoding interpretation of Corollary 16. Observe that the noise rate ε implied by Razborov-Smolensky is *above* the minimal distance

of the corresponding (punctured) Reed-Muller code of low-degree multivariate polynomials, therefore unique decoding will in general not be possible. That is, we expect many low-degree multivariate polynomials $p(x)$ to agree with a given function f_k in $\text{AC0}[\text{MOD2}]$ up to this noise rate. Identifying *any* such $p(x)$ constitutes an attack on the pseudorandomness of f_k , as it provides a low-error prediction of f_k evaluations. Note that the *number* of (punctured) Reed-Muller codewords within this distance is bounded: in particular, for $Q = 2^{\kappa^\delta}$, the probability that a random word in the space $\{0, 1\}^Q$ will be within Hamming distance $\Delta = 2^{\kappa^\delta - \log^c \kappa}$ of a codeword will be negligible. Thus, we can conclude that the existence of a weak PRF in $\text{AC0}[\text{MOD2}]$ implies that the punctured Reed-Muller code is hard to decode in some non-unique decoding regime. We formalize this in the following corollary.

Corollary 17. *Suppose for every $c, C \in \mathbb{N}$ with $c \leq C$, there is an algorithm \mathcal{A} running in time $2^{n^{o(1)}}$ such that, given a generating matrix G of a punctured RM code over \mathbb{F}_2 with parameters $(\log^C n, n)$ and corrupted codeword y , \mathcal{A} finds a codeword which is within relative distance $2^{-\log^c n}$ from y . Then there are no WPRFs in $\text{AC0}[\text{MOD2}]$.*

While it is known that the decoding of some linear codes and even structured codes such as Reed-Solomon codes for certain noise rates is NP-hard [11, 29], we are not aware of similar result for (punctured) Reed-Muller codes as the one described above. Also, to our knowledge known results on NP-hardness of computing and approximating the minimum distance of codes [25, 59] do not apply to our example. We leave it as an interesting open question to find a more natural implication from weak PRFs in $\text{AC0}[\text{MOD2}]$ to the hardness of decoding linear codes.

5.3 A Framework for VDLPN Assumptions

In [15], a candidate weak PRF in $\text{AC0}[\text{MOD2}]$ was given, with security based on a specific variable-density learning parity with noise assumption. In the following we give a framework of variable-density learning parity with noise that captures the weak PRF candidate of [15] and also our candidate based on sparse polynomials presented in Sect. 3 in $\text{AC0}[\text{MOD2}]$. Note that the VDLPN framework is not restricted to functions in $\text{AC0}[\text{MOD2}]$. And, on the other hand, not all function families in $\text{AC0}[\text{MOD2}]$ fall within this framework. Therefore, the conditional public-key implication that we give in the following only applies to candidates such as the one given in [15] and our candidate based on sparse polynomials, but not our candidate weak PRF in $\text{AC0} \circ \text{MOD2}$.

Definition 18 (A framework for VDLPN). *Let $n = n(\lambda), N = N(\lambda), \kappa = \kappa(\lambda) \in \mathbb{N}$. Let $h: \{0, 1\}^n \rightarrow \{0, 1\}^N$ and $e: \{0, 1\}^\kappa \rightarrow \{0, 1\}^N$. We say that (h, e) defines an instance of variable-density learning party with noise, if $f_k(x) := \langle h(x), e(k) \rangle$ is efficiently computable, and there exist $\zeta_i = \zeta_i(\lambda), \eta_i = \eta_i(\lambda) \in [0, 1]$ for all $i \in [N]$, such that*

1. for all $i \in [N]$ it holds: $\Pr_x[h(x)_i = 1] = \zeta_i$ and $\Pr_k[e(k)_i = 1] = \eta_i$,
2. for all $i \in [N]$ it holds: $\zeta_i \geq \zeta_{i+1}$ and $\eta_i \geq \eta_{i+1}$,
3. there exist polynomials $p = p(\lambda), q = q(\lambda) \in \mathbb{N}$ such that: $\sum_{i=1}^N \zeta_i \leq p$ and $\sum_{i=1}^N \eta_i \leq q$.

We say that the variable-density learning parity with noise (VDLPN) assumption with respect to (h, e) is (Q, T, ϵ) -hard, if $f_k(x) := \langle h(x), e(k) \rangle$ is a (Q, T, ϵ) -weak PRF.

Note that – even though not directly falling into the framework of learning parity with simple noise – VDLPN implies an instance thereof. To see this consider a VDLPN tuple (h, e) . Now, let $h_0: \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $h_1: \{0, 1\}^n \rightarrow \{0, 1\}^{N-n}$ such that $h(x) = (h_0(x), h_1(x))$ for all $x \in \{0, 1\}^n$, and similarly let $e_0: \{0, 1\}^\kappa \rightarrow \{0, 1\}^n$, $e_1: \{0, 1\}^\kappa \rightarrow \{0, 1\}^{N-n}$, such that $e(k) = (e_0(k), e_1(k))$ for all $k \in \{0, 1\}^\kappa$. Let $\mathcal{G} = \{g_k: \{0, 1\}^n \rightarrow \{0, 1\} \mid k \in \{0, 1\}^\kappa\}$, where $g_k(x) = \langle h_1(x), e_1(k) \rangle$, and let $f_{s,k} = \langle h_0(x), s \rangle \oplus g_k(x)$. Now, if VDLPN with respect to (h, e) is hard, then so is learning parity with simple deterministic noise \mathcal{G} with respect to the input distribution generated by h_0 , due to the same reduction of LPN with arbitrary secret s^* to a uniform secret s' mentioned in a comment following Corollary 16.

5.4 Connections of VDLPN to Cryptomania

In the following we outline why VDLPN “morally” implies LPN with low noise and therefore public-key encryption. We cannot show a direct PKE implication, because the Alekhovich construction [3] does not apply directly if the matrix is also sparse, since the dual LPN assumption (i.e. the assumption that the pair (H, v) for a matrix H that generates the dual code and $v = H \cdot e$ for a sparse noise vector e is indistinguishable from (H, r) for a uniformly random vector r) cannot hold true in this case, as v will be biased towards 0.

What we mean by “morally” is that the noise rate itself is sufficiently low to imply PKE, and because typically LPN is considered to be hard on average for random codes (if the noise is sufficiently dense). In order to formalize this observation we formulate a conjecture stating that if there exists a code that is hard to decode with respect to some noise rate (where the noise itself can depend on the generator matrix of the code), then “almost all” codes are hard to decode with respect to this noise rate. In order to deal with the fact that the noise might depend on the matrix (and therefore replacing the matrix might in fact trivially render LPN insecure), we simultaneously replace the noise by noise that is Bernoulli distributed at the same rate.

Conjecture 19 (Random LPN is the hardest). Let $n = n(\lambda), Q = Q(\lambda), \kappa = \kappa(\lambda), M = M(\lambda) \in \mathbb{N}$, let $\mathcal{G} = \{g_k: \{0, 1\}^n \rightarrow \{0, 1\} \mid k \in \{0, 1\}^\kappa\}$ be a family of keyed functions and let $h: \{0, 1\}^n \rightarrow \{0, 1\}^M$ such that learning parity with simple noise \mathcal{G} is (Q, T, ϵ) -hard for the input distribution generated by h . Then, we conjecture that the standard LPN problem with noise with rate η is (Q, T, ϵ) -hard. More precisely, we conjecture that if $A \xleftarrow{\$} \{0, 1\}^{Q \times M}, s \xleftarrow{\$} \{0, 1\}^M$ are both

sampled uniformly at random, and a noise vector e is sampled according to the Bernoulli distribution over $\{0, 1\}^Q$ with rate $\eta \geq \Pr_{x \xleftarrow{\$} \mathcal{D}, k \xleftarrow{\$} \{0, 1\}^\kappa} [g_k(x) = 1]$, then there exists a constant $c > 0$ such that the distribution of $(A, As + f)$ is $(T, \epsilon + 2^{-\lambda^c})$ -indistinguishable from the uniform distribution.

Note that relaxing the success probability of the adversary to $\epsilon + 2^{-\lambda^c}$ is necessary, because there obviously exist some codes that are easy to distinguish from random for any non-trivial noise rate (e.g. A chosen as the all zero matrix).

In order to further weaken the conjecture, allowing for the possibility that there exist some codes that are significantly harder to decode than random codes, one can require that the input the generated by h (i.e. obtained by sampling $x \xleftarrow{\$} \{0, 1\}^n$ and outputting $h(x)$), have min-entropy at least $\text{polylog}(\lambda)$. This weaker conjecture is still sufficient to prove the PKE implication of VDLPN.

In the full version, we prove the following.

Lemma 20. *Let $n = n(\lambda), N = N(\lambda), \kappa = \kappa(\lambda) \in \mathbb{N}$, $h: \{0, 1\}^n \rightarrow \{0, 1\}^N$ and $e: \{0, 1\}^\kappa \rightarrow \{0, 1\}^N$. Let $T = T(\lambda) \in \mathbb{N}$ and $Q = Q(\lambda)$ such that $Q \in \lambda^{\omega(1)}$. Then, if Conjecture 19 holds and VDLPN is $(Q, 2^{\lambda^c}, 2^{-\lambda^c})$ -hard for (h, e) for some constant $c > 0$, then public-key encryption with quasipolynomial running time and subexponential security exists.*

Remark 21. Note that the noise rate implied by Razborov-Smolensky does not suffice to construct public-key encryption via Alekhovich [3] (even under the “random LPN is the hardest” conjecture), because the noise rate implied by the Razborov-Smolensky approximation is $\omega(1/\sqrt{M})$. In addition, constructions of PKE from LPN with constant noise, e.g., [63], have quasi-polynomial running time and security. We are therefore not aware of any public-key implications for general weak PRFs in $\text{AC0}[\text{MOD}2]$.

Acknowledgements. We thank Andrej Bogdanov, Nicolas Resch, and the anonymous Crypto reviewers for helpful discussions and suggestions.

E. Boyle supported by ISF grant 1861/16, AFOSR Award FA9550-21-1-0046, a Google Research Award, and ERC Project HSS (852952). G. Couteau supported by the ANR SCENE. N. Gilboa supported by ISF grant 2951/20, ERC grant 876110, and a grant by the BGU Cyber Center. Y. Ishai supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and ISF grant 2774/20. L. Kohl is funded by NWO Gravitation project QSC. Research of L. Kohl was done in part while at Technion, supported by ERC Project NTSC (742754). P. Scholl supported by the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC) and an Aarhus University Research Foundation starting grant.

References

1. Akavia, A., Bogdanov, A., Guo, S., Kamath, A., Rosen, A.: Candidate weak pseudorandom functions in $AC^0 \circ MOD_2$. In: ITCS 2014. ACM, January 2014
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 430–454. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_17
3. Alekhovich, M.: More on average case vs approximation complexity. In: 44th FOCS. IEEE Computer Society Press, October 2003
4. Applebaum, B.: Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.* **42**(5), 2008–2037 (2013)
5. Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. *SIAM J. Comput.* **47**(1), 52–79 (2018)
6. Applebaum, B., Raykov, P.: Fast pseudorandom functions based on expander graphs. In: Hirt, M., Smith, A. (eds.) TCC 2016, Part I. LNCS, vol. 9985, pp. 27–56. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53641-4_2
7. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22006-7_34
8. Ball, M., Holmgren, J., Ishai, Y., Liu, T., Malkin, T.: On the complexity of decomposable randomized encodings, or: how friendly can a garbling-friendly PRF be? In: ITCS 2020. LIPIcs, January 2020
9. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_42
10. Barkol, O., Ishai, Y.: Secure computation of constant-depth circuits with applications to database search problems. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 395–411. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_24
11. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (Corresp.). *IEEE Trans. Inf. Theory* **24**(3), 384–386 (1978)
12. Blum, A., Furst, M., Kearns, M., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_24
13. Bogdanov, A., Rosen, A.: Pseudorandom functions: three decades later. *Cryptology ePrint Archive*, Report 2017/652 (2017). <http://eprint.iacr.org/2017/652>
14. Boneh, D., Ishai, Y., Passelègue, A., Sahai, A., Wu, D.J.: Exploring crypto dark matter: new simple PRF candidates and their applications. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 699–729. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_25
15. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Correlated pseudorandom functions via variable-density LPN. In: FOCS (2020)
16. Braverman, M.: Polylogarithmic independence fools AC^0 circuits. *J. ACM (JACM)* **57**(5), 1–10 (2008)
17. Canteaut, A., et al.: Stream ciphers: a practical solution for efficient homomorphic ciphertext compression. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 313–333. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_16

18. Carmosino, M.L., Impagliazzo, R., Kabanets, V., Kolokolova, A.: Learning algorithms from natural proofs. In: CCC 2016, pp. 10:1–10:24 (2016)
19. Cheraghchi, M., Grigorescu, E., Juba, B., Wimmer, K., Xie, N.: Ac^0 mod₂ lower bounds for the Boolean inner product. *J. Comput. Syst. Sci.* **97**, 45–59 (2018)
20. Courtois, N.T.: The security of hidden field equations (HFE). In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 266–281. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_20
21. Courtois, N.T.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_11
22. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_21
23. Couteau, G., Meyer, P.: Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 842–870. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_29
24. Daniely, A., Vardi, G.: From local pseudorandom generators to hardness of learning. arXiv preprint [arXiv:2101.08303](https://arxiv.org/abs/2101.08303) (2021)
25. Dumer, I., Micciancio, D., Sudan, M.: Hardness of approximating the minimum distance of a linear code. *IEEE Trans. Inf. Theory* **49**(1), 22–37 (2003)
26. Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput.* **39**(2), 606–645 (2009)
27. Filmus, Y., Ishai, Y., Kaplan, A., Kindler, G.: Limits of preprocessing. In: CCC 2020 (2020)
28. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory* **17**(1), 13–27 (1984)
29. Gandikota, V., Ghazi, B., Grigorescu, E.: On the np-hardness of bounded distance decoding of Reed-Solomon codes. In: 2015 IEEE International Symposium on Information Theory (ISIT), pp. 2904–2908. IEEE (2015)
30. Goldreich, O.: Candidate one-way functions based on expander graphs. *Cryptology ePrint Archive*, Report 2000/063 (2000). <http://eprint.iacr.org/2000/063>
31. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. IEEE Computer Society Press, October 1984
32. Håstad, J.T.: *Computational Limitations for Small-depth Circuits*. MIT Press, Cambridge (1987)
33. Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, pp. 6–20 (1986)
34. Hellerstein, L., Servedio, R.A.: On PAC learning algorithms for rich Boolean function classes. *Theor. Comput. Sci.* **384**(1), 66–76 (2007). <https://doi.org/10.1016/j.tcs.2007.05.018>
35. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC 2008, pp. 433–442 (2008)
36. Jackson, J.C.: An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.* **55**(3), 414–440 (1997)
37. Johansson, T., Jönsson, F.: Improved fast correlation attacks on stream ciphers via convolutional codes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 347–362. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_24
38. Kearns, M.J., Valiant, L.G.: Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM* **41**(1), 67–95 (1994)

39. Kharitonov, M.: Cryptographic hardness of distribution-specific learning. In: 25th ACM STOC. ACM Press, May 1993
40. Krause, M., Lucks, S.: On the minimal hardware complexity of pseudorandom function generators. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 419–430. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44693-1_37
41. Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, Fourier transform, and learnability. In: 30th FOCS. IEEE Computer Society Press, October/November 1989
42. Méaux, P., Journault, A., Standaert, F.-X., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 311–343. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_13
43. Miles, E., Viola, E.: Substitution-permutation networks, pseudorandom functions, and natural proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 68–85. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_5
44. Mossel, E., Shpilka, A., Trevisan, L.: On e-biased generators in $NC0$. In: 44th FOCS. IEEE Computer Society Press, October 2003
45. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.* **22**(4), 838–856 (1993)
46. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. In: 36th FOCS. IEEE Computer Society Press, October 1995
47. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. IEEE Computer Society Press, October 1997
48. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring (extended abstract). In: 32nd ACM STOC. ACM Press, May 2000
49. Nisan, N., Wigderson, A.: Hardness vs. randomness (extended abstract). In: 29th FOCS. IEEE Computer Society Press, October 1988
50. O’Donnell, R.: *Analysis of Boolean Functions*. Cambridge University Press, Cambridge (2014)
51. Orlandi, C., Scholl, P., Yakoubov, S.: The rise of Paillier: homomorphic secret sharing and public-key silent OT. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 678–708. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_24
52. Patarin, J.: Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt’88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_20
53. Razborov, A.A.: Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Notes Acad. Sci. USSR* **41**(4), 333–338 (1987)
54. Razborov, A.A., Rudich, S.: Natural proofs. In: 26th ACM STOC. ACM Press, May 1994
55. Rossman, B., Servedio, R.A., Tan, L.Y.: An average-case depth hierarchy theorem for Boolean circuits. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp. 1030–1048. IEEE (2015)
56. Servedio, R.A., Viola, E.: On a special case of rigidity. In: *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 19, p. 144. Citeseer (2012)
57. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: 19th ACM STOC. ACM Press, May 1987

58. Valiant, L.G.: A theory of the learnable. *Commun. ACM* **27**(11), 1134–1142 (1984)
59. Vardy, A.: The intractability of computing the minimum distance of a code. *IEEE Trans. Inf. Theory* **43**(6), 1757–1766 (1997)
60. Viola, E.: The communication complexity of addition. In: 24th SODA. ACM-SIAM, January 2013
61. Williams, R.: Natural proofs versus derandomization. In: 45th ACM STOC. ACM Press, June 2013
62. Yu, Yu., Steinberger, J.: Pseudorandom functions in almost constant depth from low-noise LPN. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 154–183. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_6
63. Yu, Yu., Zhang, J.: Cryptography with auxiliary input and trapdoor from constant-noise LPN. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 214–243. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_9



MPC-Friendly Symmetric Cryptography from Alternating Moduli: Candidates, Protocols, and Applications

Itai Dinur¹(✉), Steven Goldfeder², Tzipora Halevi³, Yuval Ishai⁴,
Mahimna Kelkar^{2,5}, Vivek Sharma⁶, and Greg Zaverucha⁷

¹ Ben-Gurion University, Beersheba, Israel
dinuri@bgu.ac.il

² Cornell Tech, New York, USA

³ Brooklyn College, CUNY, New York, USA

⁴ Technion, Haifa, Israel

⁵ Cornell University, Ithaca, USA

⁶ Graduate Center, CUNY, New York, USA

⁷ Microsoft Research, Redmond, USA

Abstract. We study new candidates for symmetric cryptographic primitives that leverage alternation between linear functions over \mathbb{Z}_2 and \mathbb{Z}_3 to support fast protocols for secure multiparty computation (MPC). This continues the study of weak pseudorandom functions of this kind initiated by Boneh et al. (TCC 2018) and Cheon et al. (PKC 2021).

We make the following contributions.

- **Candidates.** We propose new designs of symmetric primitives based on alternating moduli. These include candidate one-way functions, pseudorandom generators, and weak pseudorandom functions. We propose concrete parameters based on cryptanalysis.
- **Protocols.** We provide a unified approach for securely evaluating modulus-alternating primitives in different MPC models. For the original candidate of Boneh et al., our protocols obtain at least 2x improvement in all performance measures. We report efficiency benchmarks of an optimized implementation.
- **Applications.** We showcase the usefulness of our candidates for a variety of applications. This includes short “Picnic-style” signature schemes, as well as protocols for oblivious pseudorandom functions, hierarchical key derivation, and distributed key generation for function secret sharing.

1 Introduction

Symmetric-key cryptographic primitives, such one-way functions (OWFs) [53], pseudorandom generators (PRGs) [13, 65] and pseudorandom functions (PRFs) [39], are deployed in innumerable settings, and serve as fundamental building blocks of modern cryptography. While traditional use cases primarily considered settings where the function evaluation was done by a single party,

many applications (recently also arising in the context of cryptocurrencies) require evaluation in a distributed fashion to avoid single points of failure. This motivates the study of secure multiparty computation (MPC) protocols for evaluating such symmetric-key primitives in a setting where inputs, outputs, and keys are secret-shared or distributed between two or more parties.

Towards this goal, a long line of work [32,56,62] has made substantial progress on concretely efficient MPC protocols for distributing the computation of symmetric primitives, such as AES or SHA-256, which are widely used in practice. Unfortunately, the constructions themselves were not designed with distributed evaluation in mind, and are thus optimized for performance metrics relevant to the single-party setting. More recent work (see [3–5,15,41] and references therein) has therefore proposed to start from scratch by designing *MPC-friendly* primitives from the ground up. In this work, we continue this line of research by proposing a new suite of *simple* MPC-friendly candidate designs for a number of symmetric primitives.

Our MPC setting. We focus on the *semi-honest* setting of security for simplicity. This is considered adequate in many cases. In particular, it suffices for the construction of signature schemes via an “MPC-in-the-head” technique [25,45]. While recent general techniques from the literature [14,24] can be used to extend some of our protocols to the malicious security model with a low amortized cost, we leave such an extension to future work. We consider protocols for both two parties (2PC) and multiple parties, both with and without an honest majority assumption, and both with and without preprocessing. In the following, we consider by default the setting of (semi-honest) 2PC with preprocessing. However, our contributions apply to the other settings as well.

Efficiency metrics for MPC. Concretely efficient MPC protocols can be divided into two broad categories: protocols based on *garbled circuits* [66] and protocols based on *linear secret sharing* [10,26,40]. Protocols based on garbled circuits have low round complexity but their communication cost will be prohibitively high for our purposes. We will therefore focus on protocols based on secret sharing. Roughly speaking, the complexity of evaluating a given function f using such protocols is determined by the *size* and the *depth* of a *circuit* C that evaluates f . Here we assume that C is comprised of atomic gates of two kinds: *linear gates* (computing modular addition or multiplication by a public value) and MPC-friendly *nonlinear gates* that are supported by efficient subprotocols. A typical example for a nonlinear gate is modular multiplication of two secret values. Given such a representation for f , the *communication* cost of an MPC protocol for f scales linearly with the *size* of C , namely the number of gates weighted by the “MPC cost” of each gate, whereas the *round complexity* scales linearly with the *depth* of C , namely the number of gates on a longest input-output path. Since linear gates do not require any interaction, they do not count towards the size or the depth. We use the term “nonlinear size” and “nonlinear depth” to refer to the size and the depth when excluding linear gates.

Our design criteria. The above efficiency metrics for MPC are quite crude, since not all kinds of nonlinear gates are the same. However, they still serve

as a good intuitive guideline for the design of MPC-friendly primitives. More concretely, we would like to design primitives with the following goals in mind.

- *Low nonlinear depth.* Minimizing round complexity calls for minimizing nonlinear depth. Unfortunately, constructions like AES or even MPC-friendly ones such as LowMC [4] have quite a high nonlinear depth, which leads to high-latency protocols when using the secret-sharing approach.
- *Small nonlinear size.* For keeping the communication complexity low, we would like to minimize the number of nonlinear gates and make them as “small” and “MPC-friendly” as possible.
- *High algebraic degree.* Security of block ciphers and (weak) PRFs provably requires high algebraic degree. While there are low-degree implementations of weaker primitives such as OWFs and PRGs [6, 38, 54], these typically come at the price of bigger input size and higher nonlinear size [30, 63].
- *Simplicity.* A simple design is almost always easier to implement and prone to fewer errors and attacks. This is particularly valuable since a substantial amount of work has previously gone into implementations that resist timing and cache side-channels. Simple constructions are also easier to reason about and cryptanalyze, which builds confidence in their security, and may serve as interesting objects of study from a theory perspective [2, 38, 55].

The alternating moduli paradigm. The above design goals may seem inherently at odds with each other. How can “high algebraic degree” co-exist with “small gates” and “low nonlinear depth”? Towards settling this apparent conflict, a new design paradigm was recently proposed by Boneh et al. [15] and further explored by Cheon et al. [29]. The idea is to break the computation into two or more parts, where each part includes a linear function over a *different modulus*. The simplest choice of moduli, which also seems to lead to the best efficiency, is 2 and 3.

Boneh et al. [15] proposed a weak PRF¹ (wPRF) candidate with the following simple description: the input x is a vector over \mathbb{Z}_2 and the secret key specifies a matrix \mathbf{K} over \mathbb{Z}_2 . The PRF first computes the matrix-vector product $\mathbf{K}x$ over \mathbb{Z}_2 , then interprets the result as a vector over \mathbb{Z}_3 in the natural way, and finally applies a public, compressive linear map over \mathbb{Z}_3 to obtain an output vector y over \mathbb{Z}_3 . (When the output is a single \mathbb{Z}_3 element, the final compressive map is just a sum over \mathbb{Z}_3 .)

The above mapping from x and \mathbf{K} to y has two nonlinear steps: The first is the matrix-vector product over \mathbb{Z}_2 , whose cost can be reduced when the matrix \mathbf{K} has a special form. The second is a *conversion* of a mod-2 vector to a mod-3 vector, which consists of small (finite-size) parallel nonlinear gates. Overall, the nonlinear depth is 2. Why is this a high-degree function? Viewing both the input and the (binary representation of) the output as vectors over \mathbb{Z}_2 , high degree over

¹ A weak PRF is one whose security only holds when evaluated on *random* inputs. In many applications of strong PRF, a weak PRF can be used instead by first applying a hash function (modeled as a random oracle) to the input.

\mathbb{Z}_2 comes from the final linear map over \mathbb{Z}_3 . Viewing the input as a vector over \mathbb{Z}_3 , high degree comes from the linear map over \mathbb{Z}_2 defined by the key. Despite its simplicity, the design can be conjectured to have a good level of security with small input and key size (say, 256 bits for 128-bit security). It mostly resisted the initial cryptanalysis, where attacks found in [29] require a very big number of samples and are quite easy to circumvent by slightly modifying the design (as suggested in [29]).

A primary motivation for the alternating moduli paradigm was its MPC-friendliness. Indeed, several MPC protocols were proposed in [15]. These protocols demonstrated significant efficiency advantages over earlier MPC-friendly designs, mainly in the setting of 2PC with preprocessing or 3-party computation with an honest majority.

Another, very different, motivation is the goal of identifying simple function classes that are “hard to learn.” Indeed, the conjectures from [15] imply hardness of learning results for low complexity classes such as (depth-2) ACC⁰ circuits, sparse \mathbb{Z}_3 polynomials, or width-3 branching programs. These conjectures are also of interest outside the field of cryptography [27, 28, 36, 49], which further motivates cryptanalysis efforts.

Remaining challenges. The initial works of [15, 29] have only scratched the surface of the kind of questions one may ask.

- What about simpler symmetric primitives such as OWFs and PRGs? MPC protocols for these primitives are motivated by many applications, including Picnic-style post-quantum digital signatures [25, 50] and lightweight distributed key generation for function secret sharing [22].
- Are there similar candidates where the input, output, and key are all over \mathbb{Z}_2 ? This too is motivated by natural applications.
- Can the MPC protocols given in [15] be further improved? Can the preprocessing be realized at a low amortized cost? This motivates an additional design criterion: “PCG-friendliness,” leveraging recent advances in pseudo-random correlation generators [18, 19, 64].

1.1 Our Contributions

1.1.1 New Candidate Constructions

We introduce several candidate constructions for OWF, PRG, and (weak) PRF, all based on alternation between linear maps over \mathbb{Z}_2 and \mathbb{Z}_3 .

- **Candidate OWF.** We expand on the general structure of the (2, 3)-wPRF candidate from [15] to construct a candidate OWF. Recall that the wPRF candidate computes $\mathbf{B}(\mathbf{K}x)$ where \mathbf{K} is the secret key (over \mathbb{Z}_2) and \mathbf{B} is a compressive \mathbb{Z}_3 linear map. For our (2, 3)-OWF candidate, we replace the secret key matrix with another randomly sampled (expanding) public matrix \mathbf{A} . Specifically, given $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ and $\mathbf{B} \in \mathbb{Z}_3^{t \times m}$ where $m \geq n, t$, our OWF candidate is defined as $F(x) = \mathbf{B}(\mathbf{A}x)$ where $\mathbf{A}x$ is first reinterpreted as a 0/1 vector over \mathbb{Z}_3 .

- **Candidate wPRF.** The wPRF candidate from [15] has inputs over \mathbb{Z}_2 but outputs over \mathbb{Z}_3 . This is not suitable for applications in which the output should be further processed using secret sharing over \mathbb{Z}_2 . To this end, we propose an “LPN-style” wPRF candidate where both the input and output are over \mathbb{Z}_2 . Specifically, given a secret key matrix $\mathbf{K} \in \mathbb{Z}_2^{m \times n}$ and a public compressive map $\mathbf{B} \in \mathbb{Z}_2^{t \times m}$, for an input $x \in \mathbb{Z}_2^n$, our LPN-wPRF candidate first computes an intermediate vector

$$w = [(\mathbf{K}x \bmod 2) + (\mathbf{K}x \bmod 3) \bmod 2] \bmod 2$$

where for $\mathbf{K}x \bmod 3$, both \mathbf{K} and x are first reinterpreted over \mathbb{Z}_3 . Then, the candidate is defined as $F_{\mathbf{K}}(x) = \mathbf{B}w$. Intuitively, each intermediate vector bit can be thought of as a deterministic Learning-Parity-with-Noise (LPN) instance with a noise rate of $1/3$. The noise is deterministically generated and is dependent on the input x and a specific column of \mathbf{K} . A similar candidate was considered in [15] (as their alternate candidate) but it only outputs a single bit (it uses $\mathbf{K} \in \mathbb{Z}_2^{1 \times n}$ and outputs the intermediate vector directly). Our candidate generalizes this to multiple output bits. But more importantly, it also does not output the intermediate vector directly and instead applies an additional compressive linear map (using \mathbf{B}). We show how this allows our candidate to resist standard attacks on LPN.

- **Candidate PRG.** We also propose a candidate length-doubling PRG that is similar to our LPN-wPRF. Specifically, we use a public matrix $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ instead of the key for the first linear map. It follows the same structure as the LPN-wPRF, by first expanding the input to the intermediate vector w and then applying a compressive \mathbb{Z}_2 linear map \mathbf{B} . Choosing the length m of the intermediate vector to be large enough, we can ensure that the final compressive map still results in an output of size $t = 2n$.

1.1.2 Cryptanalysis and Implications on Parameter Choices

Algebraic attacks. Given that the constructions heavily mix linear operations over \mathbb{Z}_2 and \mathbb{Z}_3 , we will rely on the arguments of Boneh et al. [15], and conjecture that algebraic attacks do not threaten their security. Instead, we will focus on combinatorial attacks and statistical tests.

OWF. Our most interesting attack on the candidate OWF reduces the inversion problem to a particular type of subset-sum problem, where addition simultaneously involves operations over \mathbb{Z}_2 and \mathbb{Z}_3 . Thus, we can invert the OWF by applying a variant of recent subset-sum algorithms based on the *representation technique* [9, 16, 42]. Compared to a standard meet-in-the-middle approach, this attack forced us to increase the parameters by about 30%.

wPRF and PRG constructions. Our candidate constructions are related to the ones proposed in [15] and recently analyzed in [29]. The latter work describes distinguishing attacks on the constructions of [15] with asymptotically exponential (yet, concretely significant) complexity. Specifically, the attack on the (2, 3)-wPRF candidate of [15] exploits an interaction between the structure

of the circulant matrix \mathbf{K} and the choice of \mathbf{B} (which is fixed to the vector $\mathbf{1}$). On the other hand, our construction uses a random choice of \mathbf{B} which, as we show, is unlikely to result in such an interaction. The weakness in the “LPN-style” wPRF candidate of [15] was due to conditional correlation between the key and the output. We fix it by applying an additional compressive linear map.

It is important to emphasize that [29] analyzed constructions where the output length is $t = 1$, while our constructions use $t \gg 1$. Although longer output gives better performance, it may also degrade security. For example, at the extreme end, if $t = m$ the scheme is trivially broken in polynomial time by linear algebra, forcing $t \ll m$. Our security analysis shows that our candidate constructions resist such simple linear algebra attacks. Yet, the main part of security analysis is focused on statistical distinguishers that exploit a bias in the output. The strength of such a bias depends on the minimal distance of the code generated by the rows of the $t \times m$ matrix \mathbf{B} (the second linear operation of the construction). As this code is generated at random, we use the probabilistic method (in a similar way it is used to obtain the Gilbert–Varshamov bound for linear codes) to argue that its minimal distance is sufficiently large, except with negligible probability. Note that larger t results in a smaller minimal distance.

We place a concrete limit of 2^{40} on the number of samples generated by our wPRF candidates with any particular key. This reduces the probability of bad events such as collisions (where the same input to the wPRF is selected twice) and undesired interactions between the input and the structured circulant matrix \mathbf{K} . More details about such inputs are given in the security analysis.

Concrete parameters. In Table 1, we summarize the recommended concrete parameters for our constructions with the goal of obtaining s -bit security. For the (2, 3)-OWF and (2, 3)-wPRF constructions we give both aggressive and more conservative parameter sets. Note that the OWF and PRG use the minimal secret input (and output) sizes, while for wPRFs we use a larger secret. This is a result of different tradeoffs between security and performance. For example, we could have set $n = s$ for the (2, 3)-wPRF, but cryptanalysis would force setting m to be much larger than $2s$ and result in less efficient protocols. A lower bound on m in case $n = s$ is deduced by a subset-sum attack which resembles the one on the (2, 3)-OWF construction. Yet, optimizations that exploit the additional data available may be possible. While we do not expect security to degrade sharply in this case, we leave the concrete analysis for this parameter setting to future work. On the other hand, setting $n = 2s$ for the (2, 3)-OWF would also require doubling the size of the output,² once again, degrading efficiency.

Our constructions are new and it is not unlikely that some will be broken and require updating the parameter sets (even the “conservative” ones). Conversely, if for some of our constructions the more aggressive parameter sets turn out to resist future analysis, we would gain further confidence in their security.

One of the main questions we leave open is how to better exploit the structured matrices used in our constructions in cryptanalysis. This question is partic-

² Otherwise, each output would have 2^s preimages and there would be no security advantage.

Table 1. Concrete parameters for s -bit security.

Construction	Parameters (n, m, t)	Comment
(2, 3)-OWF	$(s, 3.13s, s/\log 3)$	aggressive
	$(s, 3.53s, s/\log 3)$	conservative
(2, 3)-wPRF	$(2s, 2s, s/\log 3)$	aggressive
	$(2.5s, 2.5s, s/\log 3)$	conservative
LPN-PRG	$(s, 3s, 2s)$	
LPN-wPRF	$(2s, 2s, s)$	

ularly interesting for the wPRF constructions where the attacker obtains several samples, and can perhaps utilize the structured matrices to combine their information in more efficient attacks.

1.1.3 Distributed Protocols and Optimized Implementations

As discussed above, our design criteria are guided by the goal of supporting efficient MPC protocols for distributed evaluation. We consider semi-honest protocols in several standard MPC models, either with or without preprocessing.

Efficient protocols. For our wPRF candidates, we present protocols in several different settings: (1) 2PC with preprocessing, where the input, key, and output are all secret-shared between the parties; (2) 3PC with one passive corruption, and (3) an OPRF-style 2PC with preprocessing, where one party holds the key and the other holds the input. For the (2, 3)-wPRF candidate, our 2PC protocols perform 1.5-5x better than the protocols from [15] for the same functionality, in both online communication and preprocessing size. For instance, in the 2PC setting, our protocol requires 2 rounds, 1536 bits of online communication, and 662 bits of preprocessing (i.e., correlated randomness). In contrast, the protocol from [15] for the same setting requires 4 rounds, roughly 2600 bits of online communication and roughly 3500 bits of preprocessing. Similarly, our OPRF protocol requires 2 rounds and 641 bits of online communication while the one from [15] requires 4 rounds and roughly 1800 bits of online communication.

A key ingredient for the efficiency improvement is a subprotocol for modulus conversion gates that switch between shares in \mathbb{Z}_2 and \mathbb{Z}_3 using circuit-dependent correlations. While [15] used OT in their protocols, we use these modulus conversion gates for better efficiency. We note that the same blueprint can also be used to construct efficient distributed protocols for other variants of our constructions.

Distributing the dealer at a low amortized cost. The 2PC protocols presented in [15] rely on trusted preprocessing to generate two kinds of correlated randomness. The first kind, used to securely multiply the input and the key matrix, can be thought of as a standard multiplication triple [8] over a ring. (Using a circulant matrix for the key, this involves a single multiplication in a ring of polynomials over \mathbb{Z}_2 .) It was also pointed out that using efficient pseudorandom correlation generators (PCGs) for *vector oblivious-linear evaluation* (VOLE) correlations [18, 19, 59], this kind of correlation can be generated at

a low amortized cost when the same key is reused with multiple inputs. (In fact, using more recent PCGs for independent OLE correlations [21] the latter restriction can be removed, albeit at a considerably higher cost.) The second kind of correlated randomness used in [15] is a standard oblivious transfer (OT) correlation, which can also be efficiently generated using either classical [43] or “silent” [19,64] OT extension. The latter techniques use a PCG for OT to enable fast local generation of many random instances of OT from a pair of short, correlated seeds. However, the main source of improvement over the protocols from [15] is our use of the *modulus conversion* correlations described above. We show how to generate both kinds of correlations from a standard OT correlation using only a *single* message, where in the $\mathbb{Z}_2 \rightarrow \mathbb{Z}_3$ case the (amortized) communication is < 1.38 bits per instance, and in the (less commonly used) $\mathbb{Z}_3 \rightarrow \mathbb{Z}_2$ case it is 6 bits per instance. This means that the amortized cost of distributing the dealer in our protocols is typically much lower than the cost of the online protocol that consumes the correlated randomness.

1.1.4 Applications

MPC protocols for the symmetric primitives we consider in this work are useful for a variety of cryptographic applications. Here we discuss some of these motivating applications.

Digital signatures. Using the MPC-friendliness of candidates, we can efficiently prove knowledge of an input (e.g., of an OWF input, wPRF key, or PRG seed), using proof protocols based on the MPC-in-the-head paradigm [45]. This is the approach taken by many recently designed post-quantum signature schemes [7, 11, 12, 25, 51, 58], as it only requires a secure OWF and hash function, and has opened up the range of hardness assumptions possible for public-key signatures. We present the first optimized public-key signature scheme based on alternating moduli cryptography.

We provide a detailed description of a signature scheme using our OWF candidate, as a modification to the Picnic algorithm [25, 50, 51, 61], a third round candidate in the NIST Post-Quantum Cryptography Standardization Process.³ We replace the OWF used in Picnic (an instance of the LowMC block cipher [4], which is assumed to be a OWF), update the MPC protocol accordingly, and quantify the resulting signature sizes. Using our conservative (2, 3)-OWF parameters, we find that signature sizes are slightly shorter, with signatures at the 128-bit security level (64-bit quantum) having size ranging from 10.3–13.3KB (depending on MPC parameter choices). This shows that OWFs based on alternating moduli are competitive with block-cipher based designs, with potential for future improvements, and we can choose a OWF with an (arguably) simpler mathematical description, without sacrificing performance.

Oblivious pseudorandom functions. We construct an OPRF protocol that computes our (2, 3)-wPRF candidate in an oblivious setting. In the multi-input setting (where the key is used for multiple evaluations), our protocol requires

³ See <https://csrc.nist.gov/projects/post-quantum-cryptography/>.

only 2 rounds and 641 bits of online communication. Compared to a standard DDH-based OPRF [46, 47], which require 512 bits of communication for 128-bit security, our protocol requires slightly higher communication but has a much faster online computation, which typically forms the efficiency bottleneck. In particular, our implementation shows that our OPRF protocol is faster than a *single* scalar multiplication over the Curve25519 elliptic curve. Consequently, we expect our protocol to be faster than a number of OPRF protocols [37, 48] that are based on number theoretic PRFs. Note that, unlike OPRFs based on number theoretic assumptions, ours provide plausible post-quantum security. Motivated by the latter goal, recent works [41, 60] construct an OPRF protocol from the Legendre PRF [31]. For 128-bit security and only a *single* output bit, the recent protocol from [60] has online communication cost of 13KB, substantially higher than ours (with 128 output bits), and with a higher computational cost.

Fully distributed wPRF. Unlike the OPRF setting, in which one party holds the PRF key and another holds the input, there are settings in which both the input and the key need to be distributed between two or more parties. In this setting, most of the techniques for efficient OPRF protocols (including the DDH-based protocols discussed above) do not apply. One motivating application for fully distributed wPRF, already considered in [15, 44], is a distributed implementation of *searchable symmetric encryption* (SSE) service. In distributed SSE, a client can obtain a decryption key of database entries matching a chosen keyword w by interacting with two or more servers, while keeping the keyword w secret. To this end, the client secret-shares w between the servers, who also hold shares of a wPRF key. Following interaction between the servers, the servers reveal the wPRF output to the client. This output can be used by the client to decrypt database entries associated with keyword w .

Secret-output wPRF. Our (2, 3)-wPRF candidate is well suited for applications that have privately held secret-shared inputs but require a public output that is delivered in the clear to one or more parties. However, it is insufficient for applications in which the output of the function needs to itself be kept secret and reused as the input to a subsequent PRF invocation.

One such common application arises in the context of deterministic signatures, which consists of generating a nonce by applying a PRF to the private key. In Schnorr and ECDSA, the nonce and a corresponding signature are sufficient to recover the private key. Thus, the nonce must also be distributed using the same secret-shared structure as the key. Distributed generation of deterministic signatures is once application that has both private input (the private key) and output (the nonce). Another example arises in the context of key derivation functions (KDFs), especially in a hierarchical structure, where the output of the PRF may need to be used as an input (or even a key) for another evaluation of the PRF. A related application arises in the context of Bitcoin’s BIP-32 derivation [57]. Motivated by such applications, we propose our LPN-wPRF candidate which has both its input and output over \mathbb{Z}_2 .

Distributed FSS key generation. Function secret sharing (FSS) [22] is a useful tool for a variety of cryptographic applications; see [17, 21] for recent examples.

In many of these applications, two or more parties need to securely generate FSS keys, which in turn reduces to secure evaluation of a length-doubling PRG. Our LPN-style PRG candidate serves as a good basis for such protocols. In contrast to the black-box FSS key generation protocol of Doerner and Shelat [35], its computational cost only scales logarithmically with the domain size. The optimal conjectured seed length of our PRG candidate ensures that FSS the key size is optimal as well.

1.1.5 Future Directions

Our work leaves several interesting avenues for further work. One direction is designing MPC protocols with malicious security while minimizing the extra cost. Recent techniques from [14, 24] can be helpful towards this goal. Another direction is designing and analyzing other symmetric primitives based on the alternating moduli paradigm. Relevant examples include hash functions, strong PRFs, and block ciphers. In fact, a strong PRF candidate was already suggested in [15], but analyzing its concrete security is left for future work.

2 Preliminaries

Notation. We start with some basic notation. For a positive integer k , $[k]$ denotes the set $\{1, \dots, k\}$. \mathbb{Z}_p denotes the ring of integers modulo p . We use bold uppercase letters (e.g., \mathbf{A}, \mathbf{K}) to denote matrices. We use $\mathbf{0}^l$ and $\mathbf{1}^l$ to denote the all zeros and the all ones vector respectively (of length l), and drop l when sufficiently clear. For a vector x , by $x \bmod p$, we mean that each element in x is taken modulo p . We use $x \stackrel{\$}{\leftarrow} \mathcal{X}$ to denote sampling uniformly at random a set \mathcal{X} . $\text{Funcs}[\mathcal{X}, \mathcal{Y}]$ denotes the set of all functions from \mathcal{X} to \mathcal{Y} . $a \parallel b$ denotes concatenating the strings a and b .

For distributed protocols with N parties, we use $\mathcal{P} = \{P_1, \dots, P_N\}$ to denote the set of parties. For a value x in group \mathbb{G} , we use $\llbracket x \rrbracket$ to denote an additive sharing of x (in \mathbb{G}) among the protocol parties, and $\llbracket x \rrbracket^{(i)}$ to denote the share of the i^{th} party. When clear from context (e.g., a local protocol for P_i), we will often drop the superscript. When $\mathbb{G}' = \mathbb{G}^l$ is a product group (e.g., \mathbb{Z}_p^l), for $x \in \mathbb{G}'$, we may also say that $\llbracket x \rrbracket$ is a sharing *over* \mathbb{G} , similar to the standard practice of calling x a vector over \mathbb{G} .

For a $v \in \mathbb{G}$, we use \tilde{v} to denote a random mask sampled from the same group, and $\hat{v} = v + \tilde{v}$ (where $+$ is the group operation for \mathbb{G}) to denote v masked by \tilde{v} . We use the $+$ operator quite liberally and unless specified, it denotes the group operation (e.g., component-wise addition mod p for \mathbb{Z}_p^l) for the summands.

We now briefly recall standard symmetric primitives.

Definition 1 (Weak Pseudorandom Function (wPRF)). Let $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be ensembles of finite sets indexed by a security parameter λ . Consider an efficiently computable function family $\{F_\lambda\}_{\lambda \in \mathbb{N}}$ where each function is given by $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$. We say that

$\{F_\lambda\}_{\lambda \in \mathbb{N}}$ is an (l, t, ε) -weak pseudorandom function if for infinitely many $\lambda \in \mathbb{N}$ and all adversaries \mathcal{A} running in time at most $t(\lambda)$, the following holds: taking $f_\lambda \stackrel{\$}{\leftarrow} \text{Funcs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]$, $k \stackrel{\$}{\leftarrow} \mathcal{K}_\lambda$, and $x_1, \dots, x_l \stackrel{\$}{\leftarrow} \mathcal{X}_\lambda$, we have that,

$$\left| \Pr \left[\mathcal{A} \left(1^\lambda, \{x_i, F_\lambda(k, x_i)\}_{i \in [l]} \right) \right] - \Pr \left[\mathcal{A} \left(1^\lambda, \{x_i, f_\lambda(x_i)\}_{i \in [l]} \right) \right] \right| \leq \varepsilon(\lambda).$$

Definition 2 (One-way Function (OWF)). Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be ensembles of finite sets indexed by a security parameter λ . Consider an efficiently computable function family $\{F_\lambda\}_{\lambda \in \mathbb{N}}$ where each function is given by $F_\lambda : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$. We say that $\{F_\lambda\}_{\lambda \in \mathbb{N}}$ is a (t, ε) -one-way function if for infinitely many $\lambda \in \mathbb{N}$ and all adversaries \mathcal{A} running in time at most $t(\lambda)$, we have that,

$$\Pr \left[x \stackrel{\$}{\leftarrow} \mathcal{X}; y \leftarrow F_\lambda(x) : F_\lambda(\mathcal{A}(1^{|x|}, y)) = y \right] \leq \varepsilon(\lambda)$$

Definition 3 (Pseudorandom Generator (PRG)). Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be ensembles of finite sets indexed by a security parameter λ . Consider an efficiently computable function family $\{F_\lambda\}_{\lambda \in \mathbb{N}}$ where each function is given by $F_\lambda : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$. We say that $\{F_\lambda\}_{\lambda \in \mathbb{N}}$ is an (l, t, ε) -pseudorandom generator if F is length-expanding (i.e., $\forall \lambda, \forall x \in \mathcal{X}_\lambda, |x| < |F_\lambda(x)|$) and for infinitely many $\lambda \in \mathbb{N}$ and all adversaries \mathcal{A} running in time at most $t(\lambda)$, the following holds: taking $x_1, \dots, x_l \stackrel{\$}{\leftarrow} \mathcal{X}_\lambda$ $y_1, \dots, y_l \stackrel{\$}{\leftarrow} \mathcal{Y}_\lambda$, we have that,

$$\left| \Pr \left[\mathcal{A} \left(1^\lambda, \{F_\lambda(x_i)\}_{i \in [l]} \right) \right] - \Pr \left[\mathcal{A} \left(1^\lambda, \{y_i\}_{i \in [l]} \right) \right] \right| \leq \varepsilon(\lambda).$$

3 Candidate Constructions

In this section, we introduce our suite of candidate constructions for a number of cryptographic primitives: weak pseudorandom function families (wPRF), one-way functions (OWF), and pseudorandom generators (PRG). Our constructions are all based on alternating mod-2 and mod-3 linear maps. Given the wide range of candidates we propose, we find it useful to have a clean and unified way to describe the candidate constructions in a way that will later (in Sect. 5) support a unified design of matching MPC protocols.

Circuit gates. We make use of five types of basic operations, or “gates,” which we detail below. All our constructions can be succinctly represented using just these gates. We denote by **Gates** the set comprising of these gates.

- **Mod- p Public Linear Gate.** For a prime p , given a public matrix $\mathbf{A} \in \mathbb{Z}_p^{s \times l}$, the gate $\text{Lin}_p^{\mathbf{A}}(\cdot)$ takes as input $x \in \mathbb{Z}_p^l$ and outputs $y = \mathbf{A}x \in \mathbb{Z}_p^s$.
- **Mod- p Addition Gate.** For a prime p , the gate $\text{Add}_p(\cdot, \cdot)$ takes input $x, x' \in \mathbb{Z}_p^l$ and outputs $y = x + x' \bmod p$.
- **Mod- p Bilinear Gate.** For a prime p , and positive integers s and l , the gate $\text{BL}_p^{s, l}(\cdot, \cdot)$ takes as input a matrix $\mathbf{K} \in \mathbb{Z}_p^{s \times l}$ and a vector $x \in \mathbb{Z}_p^l$ and outputs $y = \mathbf{K}x \in \mathbb{Z}_p^s$. When clear from context, we will drop the superscript and simply write $\text{BL}_p(\mathbf{K}, x)$.

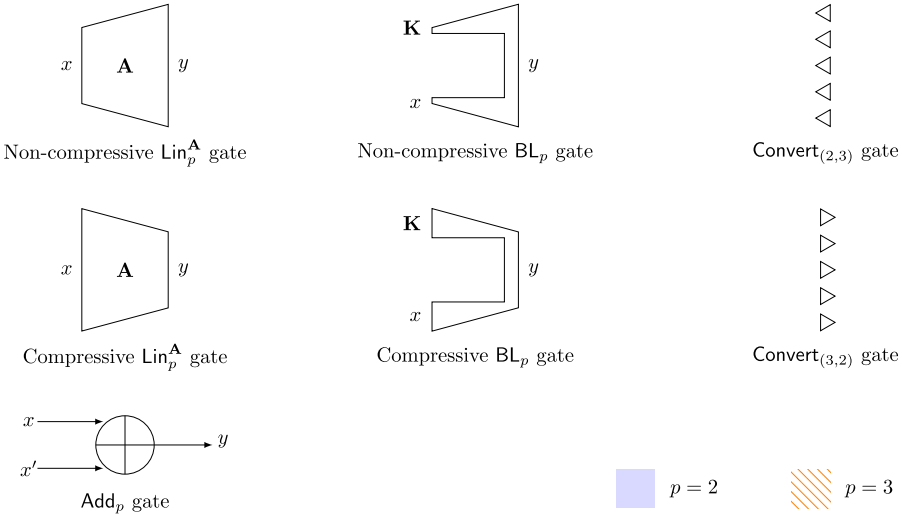


Fig. 1. Pictorial representations of the circuit gates. For the linear and bilinear gates, non-compressive means that the length of the output vector is greater than or equal to the length of the input vector, while compressive means that the output vector is smaller than the input vector. Additionally, for $p = 2$, the gates are shaded in violet, and for $p = 3$, the gates contain diagonal orange lines.

- **$\mathbb{Z}_2 \rightarrow \mathbb{Z}_3$ conversion.** For a positive integer l , the gate $\text{Convert}_{(2,3)}^l(\cdot)$ takes as input a vector $x \in \mathbb{Z}_2^l$ and returns its equivalent representation x^* in \mathbb{Z}_3^l . When clear from context, we will drop the superscript and simply write $\text{Convert}_{(2,3)}(x)$.
- **$\mathbb{Z}_3 \rightarrow \mathbb{Z}_2$ conversion.** For a positive integer l , the gate $\text{Convert}_{(3,2)}^l(\cdot)$ takes as input a vector $x \in \mathbb{Z}_3^l$ and computes its map x^* in \mathbb{Z}_2^l . For this, each \mathbb{Z}_3 element in x is computed modulo 2 to get the corresponding \mathbb{Z}_2 element in the output x^* . Specifically, each 0 and 2 are mapped to 0 while each 1 is mapped to 1. When clear from context, we will drop the superscript and simply write $\text{Convert}_{(3,2)}(x)$.

The Lin and the BL gates will behave very differently in the context of distributed protocols. For Lin, the matrix \mathbf{A} will be publicly available to all parties, while the input x will be secret shared. On the other hand, for BL, both the key \mathbf{K} and the input x will be secret shared. We call this gate *bilinear* because its output is linear in both of its (secret-shared) inputs. Also note that although the $\text{Convert}_{(2,3)}$ gate is effectively a no-op in a centralized evaluation, in the distributed setting, the gate will be used to convert an additive sharing over \mathbb{Z}_2 to an additive sharing over \mathbb{Z}_3 . Figure 1 pictorially represents each circuit gate.

Construction styles. The candidate constructions we introduce follow one of two broad styles which we detail below. A wPRF construction for the first style

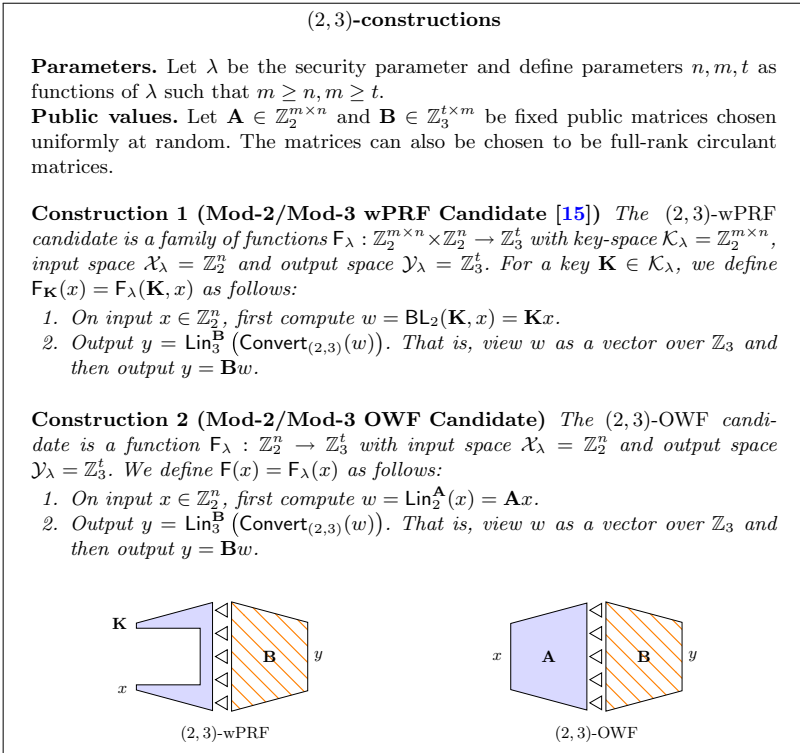


Fig. 2. (2,3)-constructions

was first proposed by [15]. Here, we also propose a suite of symmetric primitives (e.g., OWFs, PRGs) with the same basic structure.

- **(p,q)-constructions.** For distinct primes p, q , the (p, q) -constructions have the following structure: On an input x over \mathbb{Z}_p , first a linear mod p map is applied, followed by a linear mod q map. Note that after the mod p map, the input is first reinterpreted as a vector over \mathbb{Z}_q . For unkeyed primitives (e.g., OWF), both maps are public, while for keyed primitives (e.g., wPRF), the key is used for the first linear map. The construction is parameterized by positive integers n, m, t (functions of the security parameter λ) denoting the length of the input vector (over \mathbb{Z}_p), the length of the intermediate vector, and the length of the output vector (over \mathbb{Z}_q) respectively. The two linear maps can be represented by matrices $\mathbf{A} \in \mathbb{Z}_p^{m \times n}$ and $\mathbf{B} \in \mathbb{Z}_q^{t \times m}$. For keyed primitives, the key $\mathbf{K} \in \mathbb{Z}_p^{m \times n}$ will be used instead of \mathbf{A} .

Concretely, given an input $x \in \mathbb{Z}_p^n$, the construction output is of the form $y = \mathbf{B}w \in \mathbb{Z}_q^t$ where $w = \mathbf{A}x$ is first viewed over \mathbb{Z}_q . In this paper, we will analyze this style of construction for $(p, q) = (2, 3)$ and $(3, 2)$ since these are arguably the simplest constructions that employ linear maps over

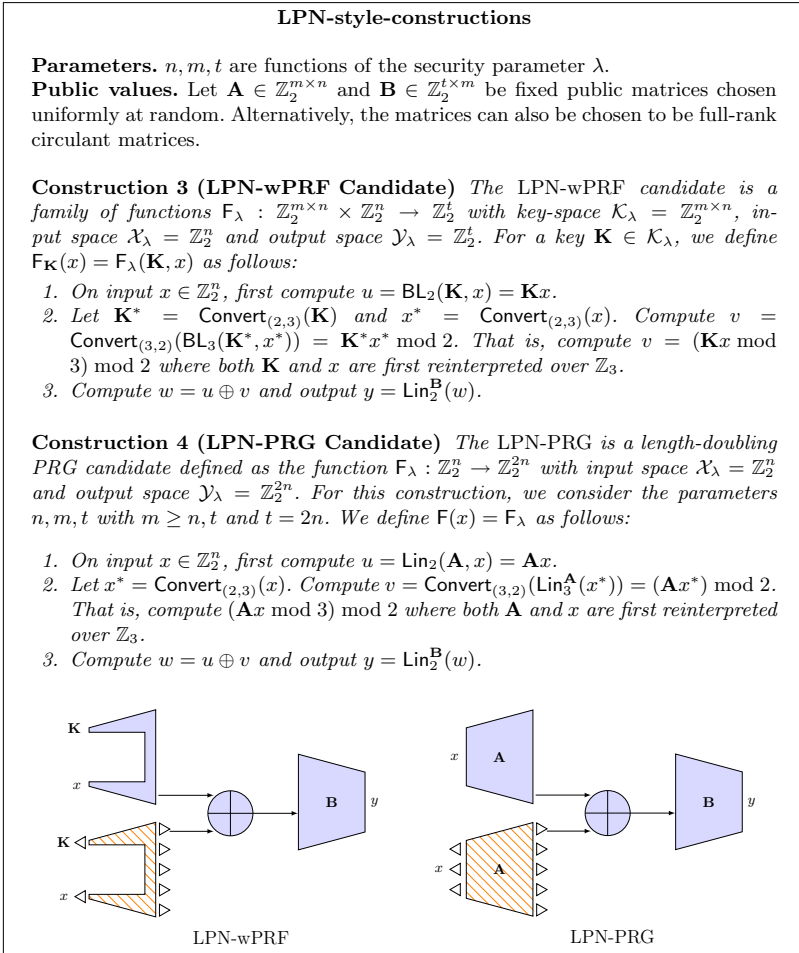


Fig. 3. LPN-style-constructions

alternate moduli. We find that the (2, 3)-constructions outperform the (3, 2)-constructions and we will primarily use the former style for our constructions. We will use (3, 2)-conversion gates in primitives where both the input *and the output* are shared over \mathbb{Z}_2 .

- **LPN-style-constructions.** These constructions have the following general structure: On input x over \mathbb{Z}_2 , first a linear mod 2 map given by the matrix \mathbf{A} is applied to obtain u . Concurrently, the same linear map is also applied over \mathbb{Z}_3 (where both x and \mathbf{A} are now reinterpreted over \mathbb{Z}_3) and then reduced modulo 2 to obtain v . The sum $w = u \oplus v$ is then multiplied by a second linear map (given by \mathbf{B}) over \mathbb{Z}_2 . The map \mathbf{B} is always public, while for keyed primitives, the key \mathbf{K} is used instead of \mathbf{A} .

The construction is parameterized by positive integers n, m, t (as functions of the security parameter λ) denoting the size of the input vector, the intermediate vector(s), and the output vector (all over \mathbb{Z}_p). Concretely, given $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ and a public $\mathbf{B} \in \mathbb{Z}_2^{t \times m}$, for an input $x \in \mathbb{Z}_2^n$, the construction first computes the intermediate vector:

$$w = [(\mathbf{A}x \bmod 2) + (\mathbf{A}x \bmod 3) \bmod 2] \bmod 2.$$

The output y is then computed as $y = \mathbf{B}w \bmod 2$. The upshot of this style is that the input and the output are both over \mathbb{Z}_2 . Intuitively, each intermediate vector bit can be thought of as a deterministic Learning-Parity-with-Noise (LPN) instance with a noise rate of $1/3$. The noise is deterministically generated and is dependent on the input x and a specific column of \mathbf{A} . The noise for the i^{th} instance will be 1 if and only if $\langle \mathbf{A}_i, x \rangle = 1$.

A similar construction was considered in [15] but only for a single-bit output. Specifically, they considered $\mathbf{A} \in \mathbb{Z}_2^{1 \times n}$ and output the single bit w . In our construction, we additionally apply a compressive linear map (using \mathbf{B}) to get the final output. This is done to resist standard attacks on LPN (see Sect. 4 for details).

Winning candidates. Through cryptanalysis and considering the cost for each candidate (See Sects. 4 and 5 for details), we find that some of our candidates are more suited (i.e., “win”) for a particular setting. Specifically, out of the candidates we consider, we find the following: (2, 3)-wPRF and (2, 3)-OWF are the best wPRF/OWF candidates with no restriction on the input/output space. LPN-wPRF is the best wPRF candidate when the input and output space are over \mathbb{Z}_2 . LPN-PRG is the best PRG candidate. We provide formal and pictorial descriptions of our winning candidates in Figs. 2 and 3.

Structured keys. The constructions we described previously use general matrices in, e.g., $\mathbb{Z}_p^{m \times n}$. For keyed primitives, this results in a key size of mn elements of \mathbb{Z}_p which is expensive to communicate within distributed protocols. Therefore, we will instead take advantage of structured matrices whose representation is only linear in n and m . Since both n and m are $O(\lambda)$ in our constructions, this reduces the communication complexity from quadratic to linear in λ . Furthermore, some structured matrices also benefit from asymptotically faster algorithms (e.g., FFT-based) for matrix multiplications and matrix-vector products. We briefly describe the types of structured matrices we utilize below. For this, consider a matrix $\mathbf{M} \in \mathbb{Z}_p^{m \times n}$.

- (Toeplitz matrices). A Toeplitz matrix, or a diagonal-constant matrix, is a matrix where each diagonal from left to right is constant. Specifically, \mathbf{M} is Toeplitz if for all $i \in [m]$ and $j \in [n]$, it holds that $M_{i,j} = M_{i+1,j+1}$ where $M_{i,j}$ denotes the element in row i and column j of \mathbf{M} . This means that a Toeplitz matrix can be represented by a single column and a single row, i.e., with $n + m - 1$ field elements.
- (Generalized circulant matrices). A generalized circulant matrix is a matrix where each row after the first, is a cyclic rotation of the first row. Specifically, if the first row of generalized circulant matrix \mathbf{M} is the vector (a_1, \dots, a_n) ,

then the m^{th} row of \mathbf{M} will be given by the same vector cyclically rotated $m-1$ times. In general, $m \neq n$, but the special case of $m = n$ is called a (square) circulant matrix. Unless specified, for brevity, we will often use the term *circulant* to denote either generalized circulant matrices or the more specific (square) circulant matrices. This will not matter for our setting, since both can be efficiently represented using just n field elements (given the dimension of the matrix).

We will usually instantiate our constructions using generalized circulant matrices to take advantage of their efficient representations. However, care must be taken while adding structure since this could potentially damage the security of a construction. Our cryptanalysis in Sect. 4 will therefore consider our constructions with structured matrices.

4 Cryptanalysis

We give a summary of cryptanalysis of our constructions, focusing on the main attacks that influence our parameters and defer details to the full version [34].

4.1 Summary of Security Evaluation of the (2, 3)-OWF

The attacker is given $\hat{y} \in \mathbb{Z}_3^t$ and tries to invert it. Our most interesting attack on the (2, 3)-OWF is based on a reduction to subset-sum.

Reduction to subset-sum. For a vector $w \in \mathbb{Z}_2^m$, there is an $(m - n) \times m$ (parity check) matrix \mathbf{P} such that there exists $x \in \mathbb{Z}_2^n$ for which $\mathbf{A}x = w$ if and only if $\mathbf{P}w = \mathbf{0}$. Assume that \hat{y} is the output of the (2, 3)-OWF on $x \in \mathbb{Z}_2^n$, and let $w = \mathbf{A}x$. Then, w satisfies the conditions $\mathbf{P}w = \mathbf{0}$ (over \mathbb{Z}_2) and $\mathbf{B}w = \hat{y}$ (over \mathbb{Z}_3). We attempt to find such w by a reduction to subset-sum, as detailed below. Suppose we find a set $J \subseteq [m]$ such that

$$\left(\sum_{j \in J} \mathbf{P}e_j \text{ mod } 2, \sum_{j \in J} \mathbf{B}e_j \text{ mod } 3 \right) = (\mathbf{0}, \hat{y})$$

where $e_i \in \{0, 1\}^m$ is the i 'th unit vector. Then, the preimage x can be computed by solving the linear equation system $\mathbf{A}x = \sum_{j \in J} e_j \text{ mod } 2$.

Thus, we have reduced the problem to subset-sum with m binary variables $(\epsilon_1, \dots, \epsilon_m) \in \{0, 1\}^m$, where we associate $\epsilon_i = 1$ with $(\mathbf{P}e_i, \mathbf{B}e_i) \in \mathbb{Z}_2^{m-n} \times \mathbb{Z}_3^t$, and define the target as $(\mathbf{0}, \hat{y}) \in \mathbb{Z}_2^{m-n} \times \mathbb{Z}_3^t$. We further note that the parity check matrix \mathbf{P} defines the linear code spanned by the columns of \mathbf{A} . Therefore, the reduction is bi-directional, implying that inverting the (2, 3)-OWF is equivalent to solving this special type of subset-sum problem.

Solving the subset-sum problem. We can now apply the advanced subset-sum algorithm by Howgrave-Graham and Joux [42] and the more recent ones [9, 16], which are based on the *representation technique*. These algorithms

were mostly designed to solve subset-sum problems over the integers. Below, we describe the main ideas of these algorithms and explain how to apply them to the special subset-sum problem we consider.

In the subset-sum problem over the integers, we are given m positive integers (a_1, a_2, \dots, a_m) and a positive integer S such that $S = \sum_{i=1}^m \epsilon_i a_i$ for $\epsilon_i \in \{0, 1\}$. The goal is to recover the unknown coefficients ϵ_i . A standard meet-in-the-middle approach for solving the problem has time complexity of about $2^{m/2}$. The representation technique gives an improved algorithm as briefly summarized below.

Assume that a solution to the subset-sum problem is chosen uniformly from $\{0, 1\}^m$ and the parameters are set such that the instance has about one solution on average. Effectively, this means that the density of the problem $d = \frac{n}{\log \max(\{a_i\}_{i=1}^m)}$ is set to 1. The main idea of the basic algorithm of Howgrave-Graham and Joux [42] is to split the problem into two parts by writing $S = \sigma_1 + \sigma_2$, where $\sigma_1 = \sum_{i=1}^m \alpha_i a_i$, $\sigma_2 = \sum_{i=1}^m \beta_i a_i$ and $(\alpha_i, \beta_i) \in \{(0, 0), (0, 1), (1, 0)\}$. Thus, $\epsilon_i = \alpha_i + \beta_i$ for each i is a solution to the problem. Note that each coefficient ϵ_i with value 1 can be split into $(0, 1)$, or $(1, 0)$. Thus, assuming that the solution has Hamming weight⁴ of $m/2$ (which occurs with probability $\Omega(1/\sqrt{m})$), it has $2^{m/2}$ different *representations*. Consequently, we may focus on finding only one of these representations by solving two subset-sum problems of Hamming weight $m/4$. Focusing on a single representation of the solution beats the standard meet-in-the-middle approach which requires time $2^{m/2}$.

Adaptation of previous subset-sum algorithms. The algorithm of [42] can be easily adapted to our specialized subset-sum problem (although it is not defined over the integers). Moreover the improved algorithm of [9] considers additional representations of the solution by allowing α_i and β_i to also take the value -1 (implying that $\epsilon_i = 0$ can be decomposed into $(\alpha_i, \beta_i) \in \{(0, 0), (-1, 1), (1, -1)\}$). In our case, we associate $\alpha_i = -1$ with $(\mathbf{P}(-e_i), \mathbf{B}(-e_i)) = (\mathbf{P}e_i, 2 \cdot \mathbf{B}e_i) \in \mathbb{Z}_2^{m-n} \times \mathbb{Z}_3^t$. Finally, the recent improved algorithm of [16] considers representations over $\{-1, 0, 1, 2\}$ and we can adapt this to our setting in a similar way. In terms of complexity, ignoring polynomial factors in m , the attack of [42] runs in time $2^{0.337m}$ and uses $2^{0.256m}$ memory, while the complexity of attack of [16] requires $2^{0.283m}$ time and memory. Thus, conservatively ignoring polynomial factors, for s -bit security we require $0.283m \geq s$, or $m \geq 3.53s$.

4.2 Summary of Security Evaluation of the (2, 3)-wPRF

For the (2, 3)-wPRF, the attacker obtains several samples $(x_1, \mathbf{B}, y_1), \dots, (x_{2r}, \mathbf{B}, y_{2r})$ and tries to mount a key recovery and/or a distinguishing attack. We restrict the number of samples produced with a single secret to 2^{40} . We set the parameters such that $n - \log 3 \cdot t \geq s$, and thus there are 2^s keys on average that are consistent with a single sample. Therefore, any key recovery attack faster

⁴ In general, one may guess the Hamming weight of the solution and repeat the algorithm accordingly a polynomial number of times.

than 2^s will use at least two samples. Particularly, the subset-sum attack can also be applied to the $(2, 3)$ -wPRF, but it is not clear how to use it efficiently on more than one sample (without strong relations between them).

The most important distinguishing attack looks for a bias in a linear combination of the output over \mathbb{Z}_3 . Given a single sample (x, \mathbf{B}, y) , assume there exist $v \in \mathbb{Z}_3^m$ and $u \in \mathbb{Z}_3^t$ such that $u\mathbf{B} = v$ and the Hamming weight of v is ℓ . As $y = \mathbf{B}w \bmod 3$, the attacker computes $uy \bmod 3 = vw \bmod 3$ and thus obtains the value of a linear combination mod 3 of ℓ entries of $w \in \{0, 1\}^m$. Since $w \in \mathbb{Z}_2^m$, this linear combination is biased, and the strength of the bias depends on how small ℓ is. The bias can be amplified using several samples. Consequently, we require that the rows of \mathbf{B} do not span a vector of low Hamming weight. This analysis is probabilistic and leads to a lower bound on m .

Another important attack we consider exploits the fact that \mathbf{K} is circulant and preserves symmetric properties of the input x (e.g., the two halves of x are equal). This attack imposes a lower bound on n so that such a symmetric vector is not found in the data, except with negligible probability. We leave it as an open problem to extend this basic attack.

Overall, we set $n = m = 2s$ and $t = s/\log 3$. These are somewhat aggressive parameters as the security margin against the above attacks is rather narrow. A choice of $n = m = 2.5s$ is more conservative.

4.3 Summary of Security Evaluation of the LPN-PRG

The attacker is given a single sample $\mathbf{A}, \mathbf{B}, y$ and tries to mount a key recovery and/or a distinguishing attack. The construction differs from the alternative wPRF construction from [15] in two ways. The first transformation generates $t = 2n$ samples using a public matrix. Similarly to [15], each sample can be viewed as an LPN sample, i.e., a noisy linear equation over \mathbb{Z}_2 in the bits of the seed (although the noise is generated deterministically). However, in [15] \mathbf{A} is a random matrix, whereas we use a (structured) Toeplitz matrix which may weaken the construction. On the other hand, the second transformation \mathbf{B} “compresses” the samples and generally strengthens the construction.

A significant consideration in selecting the parameters is that the rows of \mathbf{B} do not span a low Hamming weight vector, imposing a lower bound on m . Thus, only dense linear combinations of samples are available at the output, accumulating the noise. This should defeat standard attacks against LPN. Overall, setting $n = s, m = 3s, t = 2s$ seems to provide sufficient resistance against the considered attacks.

4.4 Summary of Security Evaluation of the LPN-wPRF

The attacks we consider against this primitive include a union of some of the attacks considered for the LPN-PRG and for the $(2, 3)$ -wPRF constructions with some adjustments. Overall, we propose to set $n = m = 2s$ and $t = s$.

5 Distributed Protocols

We now describe efficient MPC protocols to compute our candidate constructions in several useful distributed settings. First, in Sect. 5.1, we provide a technical overview for our overall protocol design. Section 5.2 quantifies this approach by providing concrete costs for distributed evaluations for our $(2, 3)$ -wPRF construction. We also provide two novel OPRF protocols based on this PRF in Sect. 5.3. In Sect. 5.4 we describe efficient protocols for distributing the generation of correlated randomness for modulus conversion gates. We defer the details of our constructions and proofs as well as protocols for other settings (3PC without preprocessing and public-input evaluation) to the full version [34].

5.1 Technical Overview

Recall that all our constructions can be succinctly represented using a set **Gates** of five basic gates. We will view each construction as a circuit over the basis **Gates** and follow the approach of [23, 33] to securely evaluate such circuits using circuit-dependent correlated randomness.

We begin with distributed protocols to evaluate each of the five gates. Abstractly, the goal of a gate protocol is to convert shares of the inputs to shares of the outputs (or shares of a masked output). To make our formalism cleaner, the gate protocols, by themselves, will involve no communication. Instead, they can additionally take in masked versions of the inputs, and possibly some additional correlated randomness. When composing gate protocols, whenever a masked input is needed, the parties will exchange their local shares to publicly reveal the masked value. This choice also prevents redoing the same communication when the masked value is already available from earlier gate evaluations.

5.1.1 Distributed Computation of Circuit Gates

We provide local protocols to compute the circuit gates we use. The description of inputs (including shared correlated randomness) and outputs for each gate protocol is also summarized in Table 2. Note that the protocols work for any number of parties. Protocols for the **Lin** and **Add** gates directly follow from the homomorphic properties of additive secret sharing, while the protocol for the **BL** gate is a generalization of Beaver’s multiplication triples [8] (see, e.g., [23]). Here, we briefly provide protocols for the new modulus conversion gates.

$\mathbb{Z}_2 \rightarrow \mathbb{Z}_3$ **conversion protocol** $\pi_{\text{Convert}}^{(2,3)}$.

- **Functionality:** Abstractly, the goal of the $\mathbb{Z}_2 \rightarrow \mathbb{Z}_3$ conversion protocol is to convert a sharing of x over \mathbb{Z}_2 to a sharing of the same $x^* = x$, but now over \mathbb{Z}_3 . For our purpose, the parties will be provided the masked input $\hat{x} = x \oplus \tilde{x}$ (i.e., masking is over \mathbb{Z}_2) directly along with correlated randomness that shares \tilde{x} over \mathbb{Z}_3 .
- **Preprocessing:** Each party is also provided with shares of the mask $r = \tilde{x}$ over \mathbb{Z}_3 as correlated randomness.

Table 2. Summary of input, output, and randomness for circuit gate protocols.

Protocol	Public Inputs	Shared Inputs	Shared Correlated Randomness	Output Shares (over base group \mathbb{G})
$\pi_{\text{Lin}}^{\mathbf{A},p}$	\mathbf{A}	x	-	$y = \mathbf{A}x$ (over \mathbb{Z}_p)
π_{Add}^p		x, x'	-	$y = x + x'$ (over \mathbb{Z}_p)
π_{BL}^p	$\hat{\mathbf{K}}, \hat{x}$	-	$\tilde{\mathbf{K}}, \tilde{x}, \tilde{\mathbf{K}}\tilde{x}$	$y = \mathbf{K}x$ (over \mathbb{Z}_p)
$\pi_{\text{Convert}}^{(2,3)}$	\hat{x} (over \mathbb{Z}_2)	-	$r = \tilde{x}$ (over \mathbb{Z}_3)	$x^* = x$ (over \mathbb{Z}_3)
$\pi_{\text{Convert}}^{(3,2)}$	\hat{x} (over \mathbb{Z}_3)	-	$u = \tilde{x} \bmod 2$ (over \mathbb{Z}_2) $v = (\tilde{x} + \mathbf{1} \bmod 3) \bmod 2$ (over \mathbb{Z}_2)	$x^* = x \bmod 2$ (over \mathbb{Z}_2)

- **Protocol details:** For the protocol $\pi_{\text{Convert}}^{(2,3)}(\hat{x} \mid r)$, each party proceeds as follows:

$$\llbracket x^* \rrbracket^{(i)} = \llbracket \hat{x} \rrbracket^{(i)} + \llbracket r \rrbracket^{(i)} + (\hat{x} \odot \llbracket r \rrbracket^{(i)}) \pmod 3$$

where \odot denotes the Hadamard (component-wise) product modulo 3.

$\mathbb{Z}_3 \rightarrow \mathbb{Z}_2$ conversion protocol $\pi_{\text{Convert}}^{(3,2)}$.

- **Functionality:** Abstractly, the goal of the protocol is to convert a sharing of x over \mathbb{Z}_3 to a sharing of $x^* = x \bmod 2$ over \mathbb{Z}_2 . For our purpose, the parties will be provided with the masked input $\hat{x} = x + \tilde{x} \bmod 3$ directly, along with correlated randomness over \mathbb{Z}_3 (see below).
- **Preprocessing:** Each party is also given shares (over \mathbb{Z}_2) of two vectors: $u = \tilde{x} \bmod 2$ and $v = (\tilde{x} + \mathbf{1} \bmod 3) \bmod 2$ as correlated randomness.
- **Protocol details:** For the protocol $\pi_{\text{Convert}}^{(3,2)}(\hat{x} \mid u, v)$, each party computes its share of x^* as follows: For each position $j \in [l]$, $\llbracket x^* \rrbracket_j^{(i)} = 1 - \llbracket u \rrbracket_j^{(i)} - \llbracket v \rrbracket_j^{(i)}$, $\llbracket v \rrbracket_j^{(i)}$, $\llbracket u \rrbracket_j^{(i)}$ when $\hat{x}_j = 0, 1, 2$ respectively.

In the full version, we show a generic technique to evaluate any construction built using the previous five gates in a distributed fashion. We also analyze the communication and preprocessing costs. Abstractly, communication will only be needed before BL, $\text{Convert}_{(2,3)}$, and $\text{Convert}_{(3,2)}$ gates to reconstruct the masked input. In terms of preprocessing, if PRG seeds are used for compression, then the computation for the $\text{BL}_p^{k,l}$, $\text{Convert}_{(2,3)}^l$, and $\text{Convert}_{(3,2)}^l$ gates will require a preprocessing of $\log_2 p \cdot k$ bits, $\log_2 3 \cdot l$ bits, and $2l$ bits respectively.

Concrete costs. In Table 3, we provide the concrete costs for our protocols in different settings for our specific parameter choices. Preprocessing costs are based on the usage of a trusted dealer. Later, in Sect. 5.4, we will show how to distribute the dealer, through efficient protocols for generating the preprocessed correlations we require from standard OT-correlations. This combined with fast silent OT [20, 64] makes the gap between the online cost mentioned in Table 3 and the *total* cost (including distributing the dealer) quite small. As a concrete example, the (amortized) total cost for the (2, 3)-wPRF in the distributed 2PC setting is only 23% higher than the online cost when a trusted dealer is used.

Table 3. Concrete MPC costs for our winning candidate constructions in three settings (Distributed 2PC (with preprocessing), 3PC, and Public-input 2PC) using our proposed parameters. For the distributed 2PC and the public-input 2PC settings, we provide the total online communication (bits, messages, rounds) and the preprocessing required in bits (without compression, with compression). For the compressed size of the preprocessing, we do not include values that can be reused (e.g., PRG seeds). For the distributed 3PC setting, we provide the total online communication cost (bits, messages, rounds) for our $(2, 3)$ -constructions. The cost of the reusable PRG seeds is not included.

Primitive	Construction	Param. (n, m, t)	Distributed 2PC (with preprocessing)		Distributed 3PC	Public-Input 2PC (with preprocessing)	
			Online Comm.	Prepr.	Online Comm.	Online Comm.	Prepr.
wPRF	$(2, 3)$ -wPRF	(256, 256, 81)	(1536, 4, 2)	(2348, 662)	(1430, 4, 1)	(512, 2, 1)	(1324, 406)
	LPN-wPRF	(256, 256, 128)	(2860, 6, 3)	(4995, 1730)		(1324, 4, 2)	(3160, 918)
OWF	$(2, 3)$ -OWF	(128, 452, 81)	(904, 2, 1)	(2337, 717)	(2525, 4, 1)	-	-
PRG	LPN-PRG	(128, 512, 256)	(1880, 4, 2)	(4334, 1227)		-	-

5.2 Distributed Evaluation in the Preprocessing Model

We briefly sketch a 2-party protocol for $(2, 3)$ -wPRF in the preprocessing model and defer details to the full version. In this setting, two parties, denoted by P_1 and P_2 hold shares of both the key \mathbf{K} and the input x . The goal is to compute shares of the output y .

For this, we provide the parties with preprocessed tuples for the BL gate, and the $\text{Convert}_{2,3}$ gate. To evaluate an input, the two parties first mask their shares of \mathbf{K} and x , and exchange them to reveal $\hat{\mathbf{K}}$ and \hat{x} . Both parties use π_{BL} to compute shares of the intermediate vector w . Then, they mask their shares and exchange them to reveal \hat{w} . The parties can now use the $\pi_{\text{Convert}}^{(2,3)}$ protocol followed by a local multiplication by \mathbf{B} to obtain shares of the output y . Note that this protocol can easily be extrapolated for distributed N -party evaluation.

5.3 Oblivious Evaluation

While our distributed protocols can be used directly for semi-honest *oblivious* PRF, or OPRF, evaluation in the preprocessing model, here we provide two protocols in this setting whose efficiency rivals that of DDH-based OPRF protocols. Recall that in the OPRF setting, one party P_1 (called the “server”) holds the key \mathbf{K} and the other party P_2 (called the “client”) holds the input x . The goal of the protocol is to have the client learn the output of the PRF for key \mathbf{K} and input x , while the server learns nothing. We provide only a brief description of our protocols next, and defer the details to the full version.

OPRF Protocol π_1^{oprf} . Our first OPRF protocol is in spirit similar to the distributed evaluation for the $(2, 3)$ -wPRF construction. Since \mathbf{K} is known to the server, and x is known to the client, both parties do not need to exchange

their shares to reconstruct the masked values $\hat{\mathbf{K}}$ and \hat{x} ; the party that holds a value can mask it locally and send it to the other party. This allows us to decouple the server’s message that masks its PRF key from the rest of the evaluation. To update the key, the server can simply send $\hat{\mathbf{K}} = \mathbf{K} + \mathbf{K}$ to the client. Many PRF evaluations can now be done using the same $\hat{\mathbf{K}}$. The upshot of this is that when the client already knows the key mask, the protocol has an optimal 2-round structure (one message from the client followed by one message from the server). For our parameters ($n = m = 256, t = 81$), π_1^{oprf} has 897 bits of online communication for input evaluation. To update the key, the server sends a 256-bit message to the client.

OPRF Protocol π_2^{oprf} . For the second protocol, the server masks the PRF in a different way; a multiplicative mask is used instead of an additive one. This saves 256 bits in the online phase at the expense of a slower key update phase.

5.4 Distributing the Trusted Dealer

In this section we show how to generate the preprocessing we require efficiently and without a trusted dealer. We will focus on the 2-party setting specifically.

5.4.1 (2, 3)-correlations from OT Correlations

We provide a new technique to generate the correlations needed for the $\pi_{\text{Convert}}^{(2,3)}$ protocol. The key technique we use is to convert OT correlations to the types of correlations our protocols require. Since prior work [19,20,64] has shown how to efficiently create OT-correlations, this implies that the correlations required for our protocols can also be efficiently generated. For a 1-out-of-2 OT correlation over \mathbb{Z}_3 , P_1 holds (z_0, z_1) and P_2 holds (c, z_c) where $z_0, z_1 \xleftarrow{\$} \mathbb{Z}_3, c \in \mathbb{Z}_2$ and $z_c = z_0$ if $c = 0$ and $z_c = z_1$ if $c = 1$. We refer to $((z_0, z_1), (c, z_c))$ as an OT correlation pair.

Conversion technique. Recall that for the $\mathbb{Z}_2 \rightarrow \mathbb{Z}_3$ conversion protocol $\pi_{\text{Convert}}^{(2,3)}$, as preprocessing, a dealer provides the parties with shares of a bit-vector both over \mathbb{Z}_2 and \mathbb{Z}_3 . For simplicity, we first consider the correlated randomness for a single element. To convert the sharing for a single bit, the dealer provides the following correlated randomness to the parties: P_1 is given (w_1, r_1) and P_2 is given (w_2, r_2) such that $w_1, w_2 \in \mathbb{Z}_2; r_1, r_2 \in \mathbb{Z}_3$ and $(w_1 + w_2) \bmod 2 = (r_1 + r_2) \bmod 3$. We refer to $((w_1, r_1), (w_2, r_2))$ as a (2, 3)-correlation pair.

We now show, in Protocol 5, how to convert an OT-correlation into a (2, 3)-correlation. Suppose for now that we have the ability to “throw” away OT-correlations where $z_0 = z_1$. We will get rid of this assumption later by communicating a single message from P_1 to P_2 which will intuitively detail which OT correlations to discard.

Protocol 5. *Given a (1-out-of-2) OT correlation $((z_0, z_1), (c, z_c))$ over \mathbb{Z}_3 where $z_0 \neq z_1$, to generate a (2, 3)-correlation, the parties proceed as follows:*

- P_1 computes

$$(w_1, r_1) = \begin{cases} (0, z_0) & \text{if } z_1 = z_0 - 1 \pmod 3 \\ (1, z_1) & \text{if } z_0 = z_1 - 1 \pmod 3 \end{cases}$$

- P_2 computes $(w_2, r_2) = (c, -z_c \pmod 3)$.

This means that an OT correlation can locally be converted to a $(2, 3)$ -correlation when $z_0 \neq z_1$. Since P_1 knows these values, it still needs to communicate to P_2 whether to use a given correlation or not. The communication can be compressed using the binary entropy function $H_b(p)$ which computes the entropy of a Bernoulli process with probability p . This leads to a communication cost of $1.5l \cdot H_b(1/3) \approx 1.377l$ for an l -length $(2, 3)$ -correlation. As another upshot, this means that the required $(2, 3)$ correlations can be generated even during the first round of the online protocol.

5.4.2 $(3, 2)$ -correlations from OT Correlations

We now show, in Protocol 6, how to convert OT-correlations to the correlations we require for the $\pi_{\text{Convert}}^{(3,2)}$ protocol. For this, we will need 1-out-of-3 OT correlations for 2-bit strings. Formally, in such a correlation, P_1 receives (z_0, z_1, z_2) where each z_j is a 2-bit string, while P_2 receives (c, z_c) where $c \in \mathbb{Z}_3$ and z_c is the corresponding z_j indexed by $j = c$. As before, these OT correlations can also be efficiently generated and compressed using existing work [20, 64].

Now, to convert a single \mathbb{Z}_2 element to \mathbb{Z}_3 , our protocol requires the following correlated randomness: P_i is given (\tilde{x}_i, u_i, v_i) where $\tilde{x}_i \in \mathbb{Z}_3$, $u_i, v_i \in \mathbb{Z}_2$ such that the following holds. Define $\tilde{x} = \tilde{x}_1 + \tilde{x} \pmod 3$, $u = u_1 + u_2 \pmod 2$, and $v = v_1 + v_2 \pmod 2$. Then, $u = \tilde{x} \pmod 2$ and $v = (\tilde{x} + 1 \pmod 3) \pmod 2$. We call this sharing between the two protocol parties a $(3, 2)$ -correlation pair.

Protocol 6. *Given a (1-out-of-3) OT-correlation $((z_0, z_1, z_2), (c, z_c))$ for 2-bit strings, to generate a $(3, 2)$ -correlation from this, the parties proceed as follows:*

- First, P_1 samples its shares randomly as $\tilde{x}_1 \xleftarrow{\$} \mathbb{Z}_3$, $u_1, v_1 \xleftarrow{\$} \mathbb{Z}_2$.
- Now, for each $j \in \mathbb{Z}_3$, P_1 sets the 2-bit string s_j as follows. Let $w = \tilde{x}_i + j \pmod 3$. Then, $s_j = (u_1 \parallel \neg v_1)$ if $w = 0$; $s_j = (\neg u_1 \parallel v_1)$ if $w = 1$; $s_j = (u_1 \parallel v_1)$ if $w = 2$. Intuitively, P_1 sets the OT tuple to be what P_2 's share would be if it chose that particular index in an OT protocol.
- P_1 masks the s_j and sends them to P_2 . Specifically, P_1 sends $r_j \leftarrow s_j + z_j$ (where each bit is added modulo 2) for each $j \in \mathbb{Z}_3$.
- P_2 sets $\tilde{x}_2 \leftarrow c$, and $u_2 \parallel v_2 \leftarrow r_c$ (i.e., the corresponding 2-bit string r_c sent by P_1 is parsed into u_2 and v_2)
- Finally, for the $(3, 2)$ -correlation, P_i takes its share as (\tilde{x}_i, u_i, v_i)

This is less efficient than generating $(2, 3)$ -correlations and takes 6 bits of communication per instance. Note that the communication is still unidirectional as only P_1 sends a message. Consequently, the $(3, 2)$ -correlations can also be generated on the fly given OT correlations as part of the first protocol round.

6 Application: Signatures with the (2, 3)-OWF

Here we describe a signature scheme using the (2, 3)-OWF. Our presentation is tailored to the (2, 3)-OWF, but we note that this approach is general. All of the candidate primitives in this paper would be a suitable choice of F (note that they are all OWFs when the input is chosen at random) and we evaluated them all before settling on (2, 3)-OWF, which gives the shortest signatures.

Abstractly, a signature scheme can be built from any OWF F and an MPC protocol to evaluate it, by setting the public key to $y = F(x)$ for a random secret x , and then proving knowledge of x , using a proof system based on the MPC-in-the-head paradigm [45]. To make the proof non-interactive, typically the Fiat-Shamir transform is used, and the message to be signed is bound to the proof by including it in the hash when computing the challenge. In addition to assuming the OWF is secure, the only other assumption required is a secure hash function. As no additional number-theoretic assumptions are required, these types of signatures are often proposed as secure post-quantum schemes.

Concretely, our design follows the Picnic signature scheme [25], specifically the variant instantiated with the KKW proof system [51] (named Picnic2 and Picnic3). We chose to use the KKW, rather than ZKB++ proof system since our MPC protocol to evaluate the (2, 3)-OWF is most efficient with a pre-processing phase, and KKW generally produces shorter signatures. We replace the LowMC block cipher [4] in Picnic with the (2, 3)-OWF, and make the corresponding changes to the MPC protocol.

This is the first signature scheme based on the hardness of inverting the (2, 3)-OWF (or similar function), a function with a simple mathematical description, making it an accessible target for cryptanalysis, especially when compared to block ciphers. Arguably, the simplicity of the OWF can lead to simpler implementations: the MPC protocol is simpler, and no large precomputed constants are required.

Our presentation is somewhat brief here as many parts are identical to Picnic. More details can be found in the full version.

Parameters. Let κ be a security parameter. The (2, 3)-OWF parameters are denoted (n, m, t) . The KKW parameters (N, M, τ) denote the number of parties N , the total number of MPC instances M , and the number τ of MPC instances where the verifier checks the online phase of simulation. The scheme also requires a cryptographic hash function.

Key generation. The signer chooses a random $x \in \mathbb{Z}_2^n$ as secret key, and a random seed $s \in \{0, 1\}^\kappa$ such that s expands to matrices $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ and $\mathbf{B} \in \mathbb{Z}_3^{m \times t}$ that are full rank (using a suitable cryptographic function, such as the SHAKE extendable output function [52]). Compute $y = F(x)$ and set (y, s) as the public key. Recall that the (2, 3)-OWF is defined as $y = F(x)$ where $x \in \mathbb{Z}_2^n$ and $y \in \mathbb{Z}_3^t$, and is computed as $y = \mathbf{B}(\mathbf{A}x)$ where $\mathbf{A}x$ is first cast to \mathbb{Z}_3 .

MPC protocol. By combining the protocols for the gates π_{Add}^3 , $\pi_{\text{Lin}}^{\mathbf{A}, 2}$, $\pi_{\text{Lin}}^{\mathbf{B}, 3}$, and $\pi_{\text{Convert}}^{(2, 3)}$ described in Sect. 5, we have an N -party protocol for the (2, 3)-

Table 4. Signature size estimates for Picnic using $(2, 3)$ -OWF, compared to Picnic using LowMC. The left table shows security level L1 (128 bits) with $N = 16$ and $N = 64$ parties, and the right table shows level L5 (256 bits).

OWF Params (n, m, t)	KKW params (N, M, τ)	Sig. size (KB)	OWF Params (n, m, t)	KKW params (N, M, τ)	Sig. size (KB)
(128, 453, 81)	(16, 150, 51)	13.30	(256, 906, 162)	(16, 324, 92)	50.19
	(16, 168, 45)	12.48		(16, 400, 79)	47.08
	(16, 250, 36)	11.54		(16, 604, 68)	45.82
Picnic3-L1	(16, 250, 36)	12.60	Picnic3-L5	(16, 604, 68)	48.72
(128, 453, 81)	(64, 151, 45)	13.59	(256, 906, 162)	(64, 322, 82)	51.23
	(64, 209, 34)	11.70		(64, 518, 60)	44.04
	(64, 343, 27)	10.66		(64, 604, 57)	43.45
Picnic2-L1	(64, 343, 27)	12.36	Picnic2-L5	(64, 604, 58)	46.18

OWF. The most challenging and costly step (in terms of communication) is the conversion gate, all other operations are done locally by the parties.

Sign and verify. The prover simulates the preprocessing and online phase for all M MPC instances, and commits to the preprocessing values, and MPC inputs and outputs. Then she is challenged to open τ of the M MPC instances. The verifier will check the simulation of the online phase for these instances, by recomputing all values as the prover did for $N - 1$ of the parties, and for remaining unopened party, the prover will provide the missing broadcast messages and commitments so that the verifier may complete the simulation and recompute all commitments. For the $M - \tau$ instances not chosen by the challenge, the verifier will check the preprocessing phase only, by recomputing the preprocessing phase as the prover did.

Parameter selection and signature size. The impact of OWF choice is limited to one term, which is the sum of the sizes of the MPC inputs, broadcast messages, and auxiliary values produced by preprocessing. Selecting the KKW parameters (M, N, τ) once the MPC costs are known follows the approach in Picnic: a range of options are possible, and we try to select parameters that balance speed (mostly dependent on the number of MPC executions and number of parties) and size. Since the MPC costs of the $(2, 3)$ -OWF are very close to those of LowMC, the options follow a similar curve.

Table 4 gives some options with $N = 16, 64$ parties, providing 128 and 256 bits of security. For each category, we highlight the row of $(2, 3)$ -OWF parameters that are a direct comparison to Picnic. Signatures using the $(2, 3)$ -OWF are slightly shorter (five to fifteen percent) than Picnic using LowMC.

7 Implementation and Evaluation

We implemented our 2-party protocols to compute the $(2, 3)$ -wPRF candidate (Construction 1) both in the distributed and oblivious evaluation settings. Our

Table 5. Centralized 23-wPRF benchmarks for a baseline implementation and for different optimization techniques. Packing was done into 64-bit sized words (for both \mathbb{Z}_2 and \mathbb{Z}_3 vectors). For the lookup table optimization, a table with 81×2^{20} \mathbb{Z}_3 elements, or roughly of size 135MB, was preprocessed. Runtimes are all given in microseconds (μs).

Optimization			Runtime (μs)	Evaluations/sec
Packing	Bit Slicing	Lookup Table		
Baseline implementation			156.41	6K
✓			26.84	37K
✓	✓		18.5	65K
✓	✓	✓	6.08	165K

implementations are in C++. For the (2, 3)-wPRF construction, we used the parameters $n = m = 256$ and $t = 81$. The implemented 23-constructions use a Toeplitz matrix in $\mathbb{Z}_2^{256 \times 256}$ as the key, take as input a vector in \mathbb{Z}_2^{256} and output a vector in \mathbb{Z}_3^{81} . The correlated randomness was implemented as if provided by a trusted third party. See Sect. 5.4 for concretely efficient protocols for securely generating the correlated randomness, which we did not implement but give efficiency estimates based on prior works.

Optimizations. We start with a centralized implementation of the 23-wPRF. We find optimizations that provide a roughly 25x better performance over a naïve implementation. We use three major optimizations in our implementation. First, we use *bit packing* for \mathbb{Z}_2 vectors through which we can pack several elements in a machine word and operate on them together in an SIMD manner. Second, we use *bit slicing* for \mathbb{Z}_3 vectors by representing them as a pair of \mathbb{Z}_2 vectors. All operations on the \mathbb{Z}_3 vectors can now be translated to operations on the \mathbb{Z}_2 vectors. Finally, we use a lookup table optimization for the final \mathbb{Z}_3 linear mapping (i.e., multiplication by \mathbf{B}). For this, we split the 256-column matrix \mathbf{B} into 16 pieces with 16 columns each and store multiplications with all \mathbb{Z}_3^{16} vectors for each piece. The size for each piece was decided as a tradeoff between the lookup table size and the computational efficiency. We provide benchmarks for our optimizations in Table 5.

7.1 Performance Benchmarks

Experimental setup. We ran all our experiments on a t2.medium AWS EC2 instance with 4GiB RAM (architecture: x86-64 Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz) running on Ubuntu 18.04. The performance benchmarks and timing results we provide are averaged over 1000 runs. For the distributed construction benchmarks, both parties were run on the same instance. We separately report the computational runtime for the parties, and analytically compute the communication costs.

Table 6. Comparison of protocols for (semi-honest) OPRF evaluation in the preprocessing model. Runtimes in microseconds (μs) are provided separately for refreshing the key (Key Update) and for evaluating an input (Evaluation). Communication and preprocessing are also provided separately for the two stages.

Protocol		Runtime (μs)		Preprocessing (bits)	Communication (bits)	
		Client	Server		Client	Server
π_1^{oprf}	Key Update	-	0.65	256	-	256
	Evaluation	8.54	9.45	2092	512	385
π_2^{oprf}	Key Update	-	3.16	256	-	256
	Evaluation	7.91	8.21	1836	256	385
DDH-based OPRF		57.38	28.69	-	256	256

Distributed wPRF evaluation. We implement our 2-party semi-honest distributed protocol for evaluating the $(2, 3)$ -wPRF construction and report timings for our implementation. Since this candidate was first proposed in [15], we also implement their protocol as a comparison point. For both protocols, we use the parameters $n = m = 256$, $t = 81$ for the PRF and use the same optimizations for an accurate comparison. We found that our protocol is better in all metrics. For a single evaluation, our protocol requires $12.12 \mu s$, 662 bits of preprocessing, and 1536 bits of online communication. On the other hand, the protocol from [15] requires $28.02 \mu s$, 3533 bits of preprocessing, and 2612 bits of online communication for one evaluation.

OPRF evaluation. In Table 6, we provide performance benchmarks for both our oblivious protocols (see Sect. 5.3) for the $(2, 3)$ -wPRF construction. We also compare our results to the standard DDH-based OPRF (details in the full version [34]). For our timing results, we report both the server and client runtimes (averages over 1000 runs). For each construction, we also include the size of the preprocessed correlated randomness, and the online communication cost. All constructions are parameterized appropriately to provide 128-bit security.

For our constructions, we report separately, the timings for refreshing the key and evaluating the input. For the comparison with the DDH-based OPRF construction, we use the libsodium library [1] for the elliptic curve scalar multiplication operation. We use the Curve25519 elliptic curve, which has a 256-bit key size, and provides 128 bits of security.

Acknowledgments. Itai Dinur is supported by ISF grants 573/16 and 1903/20, and by the European Research Council under the ERC starting grant agreement No. 757731 (LightCrypt). Yuval Ishai is supported by ERC Project NTSC (742754), ISF grant 2774/20, NSF-BSF grant 2015782, and BSF grant 2018393.

References

1. libsodium 1.0.18-stable (2020). <https://libsodium.gitbook.io/doc/>. Accessed 31 Dec 2020
2. Akavia, A., Bogdanov, A., Guo, S., Kamath, A., Rosen, A.: Candidate weak pseudorandom functions in AC MOD 2. In: ITCS, pp. 251–260 (2014)
3. Albrecht, M.R., et al.: Feistel structures for MPC, and more. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11736, pp. 151–171. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29962-0_8
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_17
5. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. TOSC **2020**(3), 1–45 (2020)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0 . In: FOCS, pp. 166–175 (2004)
7. Baum, C., de Saint Guilhem, C.D., Kales, D., Orsini, E., Scholl, P., Zaverucha, G.: Banquet: Short and fast signatures from AES. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12710, pp. 266–297. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75245-3_11
8. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
9. Becker, A., Coron, J.-S., Joux, A.: Improved generic algorithms for hard knapsacks. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 364–385. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_21
10. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10 (1988)
11. Beullens, W.: Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 183–211. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_7
12. Beullens, W., Delpech de Saint Guilhem, C.: LegRoast: Efficient post-quantum signatures from the legendre PRF. In: Ding, J., Tillich, J.-P. (eds.) PQCrypto 2020. LNCS, vol. 12100, pp. 130–150. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_8
13. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudorandom bits. SICOMP **13**(4), 850–864 (1984)
14. Boneh, D., Boyle, E., Corrigan-Gibbs, H., Gilboa, N., Ishai, Y.: Zero-knowledge proofs on secret-shared data via fully linear PCPs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 67–97. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_3
15. Boneh, D., Ishai, Y., Passelègue, A., Sahai, A., Wu, D.J.: Exploring crypto dark matter: New simple PRF candidates and their applications. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 699–729. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_25
16. Bonnetain, X., Bricout, R., Schrottenloher, A., Shen, Y.: Improved classical and quantum algorithms for subset-sum. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 633–666. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_22

17. Boyle, E., et al.: Function secret sharing for mixed-mode and fixed-point secure computation. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 871–900. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_30
18. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: CCS, pp. 896–912 (2018)
19. Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: CCS, pp. 291–308 (2019)
20. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudo-random correlation generators: silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 489–518. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_16
21. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12171, pp. 387–416. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_14
22. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_12
23. Boyle, E., Gilboa, N., Ishai, Y.: Secure computation with preprocessing via function secret sharing. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11891, pp. 341–371. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_14
24. Boyle, E., Gilboa, N., Ishai, Y., Nof, A.: Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In: CCS, pp. 869–886 (2019)
25. Chase, M., et al.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: CCS, pp. 1825–1842 (2017)
26. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC, pp. 11–19 (1988)
27. Chen, L.: Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In: FOCS, pp. 1281–1304 (2019)
28. Chen, L., Ren, H.: Strong average-case lower bounds from non-trivial derandomization. In: STOC, pp. 1327–1334 (2020)
29. Cheon, J.H., Cho, W., Kim, J.H., Kim, J.: Adventures in crypto dark matter: Attacks and fixes for weak pseudorandom functions. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 739–760. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75248-4_26
30. Couteau, G., Dupin, A., Méaux, P., Rossi, M., Rotella, Y.: On the concrete security of Goldreich’s pseudorandom generator. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 96–124. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_4
31. Damgård, I.B.: On the randomness of Legendre and Jacobi sequences. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 163–172. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_13
32. Damgård, I., Keller, M.: Secure multiparty AES. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 367–374. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14577-3_31
33. Damgård, I., Nielsen, J.B., Nielsen, M., Ranellucci, S.: The TinyTable Protocol for 2-party secure computation, or: gate-scrambling revisited. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 167–187. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_6

34. Dinur, I., et al.: MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. *Cryptology ePrint Archive*, report number 2021/885 (2021). <https://eprint.iacr.org/2021/885.pdf>
35. Doerner, J., Shelat, A.: Scaling ORAM for secure computation. In: *CCS*, pp. 523–535 (2017)
36. Filmus, Y., Ishai, Y., Kaplan, A., Kindler, G.: Limits of preprocessing. In: *CCC*, pp. 17:1–17:22 (2020)
37. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) *TCC 2005*. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_17
38. Goldreich, O.: Candidate one-way functions based on expander graphs. In: Goldreich, O. (ed.) *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. LNCS, vol. 6650, pp. 76–87. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22670-0_10
39. Goldreich, O., Goldwasser, S., Micali, S.: On the cryptographic applications of random functions (extended abstract). In: Blakley, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 276–288. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_22
40. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: *STOC*, pp. 218–229 (1987)
41. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-friendly symmetric key primitives. In: *CCS*, pp. 430–443 (2016)
42. Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. In: Gilbert, H. (ed.) *EUROCRYPT 2010*. LNCS, vol. 6110, pp. 235–256. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_12
43. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) *CRYPTO 2003*. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
44. Ishai, Y., Kushilevitz, E., Lu, S., Ostrovsky, R.: Private large-scale databases with distributed searchable symmetric encryption. In: Sako, K. (ed.) *CT-RSA 2016*. LNCS, vol. 9610, pp. 90–107. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_6
45. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: *STOC*, pp. 21–30 (2007)
46. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014*. LNCS, vol. 8874, pp. 233–253. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_13
47. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: Highly-efficient and composable password-protected secret sharing (or: How to protect your Bitcoin wallet online). In: *EURO S&P*, pp. 276–291 (2016)
48. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) *TCC 2009*. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_34
49. Kabanets, V., Koroth, S., Lu, Z., Myrasiotis, D., Oliveira, I.: Algorithms and lower bounds for De Morgan formulas of low-communication leaf gates. In: *CCC*, pp. 15:1–15:41 (2020)
50. Kales, D., Zaverucha, G.: Improving the performance of the Picnic signature scheme. *TCHES* **2020**(4), 154–188 (2020)

51. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: CCS, pp. 525–537 (2018)
52. Kelsey, J., Chang, S.J., Perlner, R.: SHA-3 derived functions: cSHAKE KMAC TupleHash and ParallelHash. National Institute for Standards and Technology, Special Publication 800-185 (2016)
53. Levin, L.: One-way functions and pseudorandom generators. In: STOC, pp. 363–365 (1985)
54. Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: Barstow, D., et al. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-45961-8_39
55. Miles, E., Viola, E.: Substitution-permutation networks, pseudorandom functions, and natural proofs. *J. ACM* **62**(6), 46:1–46:29 (2015)
56. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_15
57. Proposal, B.I.: Hierarchical deterministic wallets (2017). https://en.bitcoin.it/wiki/BIP_0032
58. de Saint Guilhem, C.D., De Meyer, L., Orsini, E., Smart, N.P.: BBQ: using AES in Picnic signatures. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 669–692. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38471-5_27
59. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: improved constructions and implementation. In: CCS, pp. 1055–1072 (2019)
60. Seres, I.A., Horváth, M., Burcsi, P.: The Legendre pseudorandom function as a multivariate quadratic cryptosystem: security and applications. Cryptology ePrint Archive, Report 2021/182 (2021)
61. The Picnic Design Team: The Picnic signature algorithm specification, version 3.0, September 2020. <https://microsoft.github.io/Picnic/>
62. Wang, X., Ranellucci, S., Katz, J.: Global-scale secure multiparty computation. In: CCS, pp. 39–56 (2017)
63. Yang, J., Guo, Q., Johansson, T., Lentmaier, M.: Revisiting the concrete security of Goldreich’s pseudorandom generator (2021)
64. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated OT with small communication. In: CCS, pp. 1607–1626 (2020)
65. Yao, A.C.: Theory and application of trapdoor functions. In: FOCS, pp. 80–91 (1982)
66. Yao, A.C.: How to generate and exchange secrets. In: FOCS, pp. 162–167 (1986)



No Time to Hash: On Super-Efficient Entropy Accumulation

Yevgeniy Dodis¹, Siyao Guo^{2(✉)}, Noah Stephens-Davidowitz³, and Zhiye Xie²

¹ New York University, New York, NY, USA
dodis@cs.nyu.edu

² New York University Shanghai, Shanghai, China
{siyao.guo, zx572}@nyu.edu

³ Cornell University, Ithaca, NY, USA
noahsd@gmail.com

Abstract. Real-world random number generators (RNGs) cannot afford to use (slow) cryptographic hashing every time they refresh their state R with a new entropic input X . Instead, they use “superefficient” simple entropy-accumulation procedures, such as

$$R \leftarrow \text{rot}_{\alpha,n}(R) \oplus X,$$

where $\text{rot}_{\alpha,n}$ rotates an n -bit state R by some fixed number α . For example, Microsoft’s RNG uses $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$. Where do these numbers come from? Are they good choices? Should rotation be replaced by a better permutation π of the input bits?

In this work we initiate a rigorous study of these pragmatic questions, by modeling the sequence of successive entropic inputs X_1, X_2, \dots as *independent* (but otherwise adversarial) samples from some natural distribution family \mathcal{D} . Our contribution is as follows.

- We define *2-monotone distributions* as a rich family \mathcal{D} that includes relevant real-world distributions (Gaussian, exponential, etc.), but avoids trivial impossibility results.
- For any α with $\gcd(\alpha, n) = 1$, we show that rotation accumulates $\Omega(n)$ bits of entropy from n independent samples X_1, \dots, X_n from any (unknown) 2-monotone distribution with entropy $k > 1$.
- However, we also show some choices of α perform much better than others for a given n . E.g., we show $\alpha = 19$ is one of the best choices for $n = 64$; in contrast, $\alpha = 5$ is good, but generally worse than $\alpha = 7$, for $n = 32$.
- More generally, given a permutation π and $k \geq 1$, we define a simple parameter, the *covering number* $C_{\pi,k}$, and show that it characterizes the number of steps before the rule

$$(R_1, \dots, R_n) \leftarrow (R_{\pi(1)}, \dots, R_{\pi(n)}) \oplus X$$

accumulates nearly n bits of entropy from independent, 2-monotone samples of min-entropy k each.

- We build a simple permutation π^* , which achieves nearly optimal $C_{\pi^*,k} \approx n/k$ for all values of k *simultaneously*, and experimentally validate that it compares favorably with all rotations $\text{rot}_{\alpha,n}$.

Keywords: Randomness extractors · Entropy accumulation · RNGs

1 Introduction

Good random number generation is essential for cryptography and beyond. In practice, this difficult task is solved by a primitive called a *Random Number Generator* (*RNG*, or *RNG with input*), whose aim is to quickly accumulate entropy from various physical entropic sources in the environment with unknown distributions (such as timing of interrupts, etc.). The RNG then converts this high-entropy state into the (pseudo)random bits that are needed for applications. In this work we focus on the first step: *entropy accumulation*. This is usually achieved by a procedure $S \leftarrow \text{Refresh}(S, X)$, where S is the state of the RNG, and X is the entropic input whose entropy we are trying to “accumulate” into the fixed-length RNG state S .¹ Intuitively, we wish to design Refresh so that S converges to a high-entropy, and eventually (almost) uniform distribution, provided that the input samples X_1, X_2, \dots collectively have enough entropy, without too many additional assumptions about the X_i .

In the context of RNGs, the requirement of entropy accumulation was formalized by Dodis et al. [9] (building on prior influential work of [3]), and there has been much follow-up work [7, 10, 14, 16]. Most of these works consider a very powerful adversary, who tries to choose the *worst* possible entropy source for the Refresh subject to satisfying the overall entropy constraints. As such, all existing Refresh procedures in the literature are relatively expensive, using either a cryptographic hash function Hash which simply sets $S \leftarrow \text{Hash}(S, X)$ for the new input X , or, under some additional assumptions [9], a full field multiplication over a finite field $\text{GF}[2^N]$ for large values of N (on the order of 500–1000).

Unfortunately, the Refresh procedures from these theoretical works appear to miss the following critical consideration, making them insufficient for real-world RNG design. Many practical entropy sources—such as interrupt timings—come at a very rapid pace (but possibly with relatively low entropy per sample). Hence, running a cryptographically secure hash function (or doing a very large finite field multiplication) every time we receive such an input X would be not only be prohibitively expensive, but *completely infeasible* for an operating system RNG, for example.

As a result, practitioners use the following elegant compromise, not yet modeled by the theory of RNGs (prior to our work), but found in every major operating system including `/dev/random` [23] for Linux, Yarrow [19] for MacOS/iOS/FreeBSD, and Fortuna [13] for Windows [11, 12]. The state of the RNG will consist of two pieces: a relatively long state S for the “slow” entropy accumulation procedure we denote by Slow-Refresh, and a small array of very short states R —sometimes called *registers*—for the “superefficient” entropy accumulation procedure Fast-Refresh. On every single interrupt timing X , one

¹ Equivalently, one can think of the refresh procedure as a *randomness condenser* [21, 22], which condenses $|S| + |X|$ bits back to $|S|$ bits, while trying not to lose the overall entropy in *both* S and X (and therefore “accumulating” the fresh entropy brought by X back into the state S).

always updates one of the registers R (usually in some round-robin manner):

$$R \leftarrow \text{Fast-Refresh}(R, X)$$

Since interrupts could happen very frequently, the mandatory requirement for the fast refresh operation is *extreme speed and simplicity*. We comment on this below, but first complete the refresh procedure description. Less frequently, one would accumulate the state of all the registers $\{R\}$ into the long RNG state S :

$$S \leftarrow \text{Slow-Refresh}(S, \{R\})$$

The latter function is typically implemented as a cryptographic hash function Hash , and can afford to be much slower. It is then this longer state S that will be used to generate (pseudo)random bits. This in particular means that the registers R do not need to achieve the same guarantees as the larger state S .

All existing theoretical modelings of RNGs with input only focused on the slow accumulation procedure Slow-Refresh . As such, it completely abstracted away a key question concerning the design of all practical RNGs:

What is the best way to design extremely fast and practical refresh procedures Fast-Refresh to accumulate entropy as fast as possible?

The goal of this work is to model these super-efficient entropy accumulators, and to build the theoretical foundations for this very important primitive. Hence, for much of this work (with the exception of Sect. 8), we will completely ignore Slow-Refresh (and all other details of RNGs), and focus exclusively on the clear question of understanding the design of super-efficient entropy accumulation.

Existing Designs: Cyclic Rotation. To dig into our question a bit deeper, it is helpful to see what is typically done in practice. As we said, the fast-refresh procedure has to be blazing fast, and can realistically involve just a few simple bit-level operations applied to the entropic input and the register. In fact, most RNGs we know, such as the one used by Windows 10 [12], implement the following “rotate-then-xor” procedure. The register R is typically an $n = 32$ or $n = 64$ -bit value. The raw input X —such as the timing of the previous interrupt—is also an n -bit string. To refresh the register (quickly!), one simply cyclically rotates the bits of the register by some fixed constant α (e.g., rotation by two would map the bit string $(1, 1, 0, 1, 0, 1)$ to $(0, 1, 1, 1, 0, 1)$), and then XORs the input X to the result:

$$R \leftarrow \text{rot}_{\alpha, n}(R) \oplus X$$

Concretely, Microsoft uses $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$ [12].

Our Questions. While this design appears reasonable, it raises a lot of questions that we would like to answer.

- How were these (seemingly mysterious) rotation numbers α selected?

- Is there some rigorous metric/model that can help compare different rotation amounts to each other, either practically, or theoretically, or both?
- In particular, are Microsoft’s choices of $\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$ “good”?
- How should one model the distributions of the entropic inputs X to properly study these refresh procedures?
- Is rotation really the best way to permute the bits of the state for entropy accumulation?
- In particular, can rotation be replaced by a “better” permutation π of the n register positions: $(R_1, \dots, R_n) \leftarrow (R_{\pi(1)}, \dots, R_{\pi(n)}) \oplus X$?

To start answering these questions informally, let us make some simple observations to get some intuition for why Microsoft might have chosen these seemingly mysterious numbers, 5 and 19. First, it seems clear that we should take the rotation amount α relatively prime to n , to make sure every bit eventually affects every other bit. Second, we claim that it is unwise to take α very small (e.g., $\alpha = 1$), since practical sources will tend to “have most of their entropy in the lower-order bits,” so that small values of α will take a lot of time to affect all the bits of the register. For example, even if every sample of X is uniform in its $n/2$ least significant bits, rotation by 1 will only accumulate $n/2 + \ell - 1$ bits after ℓ steps. For a similar reason, one should avoid values of α where a *small multiple* of α is very close to n ; such as $\alpha = 11$ for $n = 32$, or $\alpha = 21$ for $n = 64$. For example, after three such steps with $\alpha = 11$ for $n = 32$, a fresh sample which is uniform in its 5 least-significant bits will contribute only one fresh bit of entropy, just as if we had $\alpha = 1$.

Choosing between the remaining possibilities of, e.g., $\alpha = 5, 7, 9, \dots$, yields subtle tradeoffs. Indeed, while it is clear that there is something interesting going on here, it is not immediately clear how to formalize this.

1.1 Our Model

In this work, we use the tools of modern information theory and cryptography to make the above ad-hoc arguments more systematic and theoretically sound, so that we have higher confidence in the quality of our answers. In the process, we will uncover some interesting theory.

Syntax and Efficiency. First, we restrict our attention to entropy accumulation of the form

$$(R_1, \dots, R_n) \leftarrow (R_{\pi(1)}, \dots, R_{\pi(n)}) \oplus X$$

for some permutation $\pi : [n] \rightarrow [n]$ of the n -bit register R , as this model is quite natural in our context of super-efficient constructions. For conciseness, we let $A_\pi(R) = (R_{\pi(1)}, \dots, R_{\pi(n)})$, with cyclic rotation $\text{rot}_{\alpha, n}(R)$ being of most immediate interest to us.

Modeling of Entropic Inputs X . Given the extreme simplicity of our accumulation procedure, it is clear that we will not be able to withstand the same level

of generality and “malicious” attacks that are modeled in prior work addressing the complementary question of “slow refresh”. For example, even if the marginal distributions of X_i are completely uniform in $\{0, 1\}^n$, we will fail to accumulate a single bit of entropy if, for example, the X_i satisfy $A_\pi(X_{2i-1}) = X_{2i}$. (The state will be zero after every even number of steps when starting from $R = 0^n$!)

Hence, as the first modeling assumption we will assume that the inputs X_i are *independent*. This is a common abstraction in the randomness extraction literature dating back to [6].² While it might not be entirely accurate in practice, we believe that it captures some of what is useful about natural sources such as interrupt timings, which do not appear fully adversarial.

Second, to minimize the number of parameters, and also to focus on the high-level picture, in our analyses we will assume that the entropy of each (independent) sample is lower bounded by some parameter k . (Once again, this is standard in the randomness extraction literature; with very few exceptions, such as [17].) The key point is that our refresh procedure does not know/use anything about k , and a “good” result should yield quick entropy accumulation for *all values* of k ; presumably in roughly n/k steps, which is the best possible. Thus, even if the quality of source is unknown, a “good” result of this type will tell us that our entropy accumulation always works to the best extent possible.

Finally, we will further restrict each sample to come from some (natural) family of distributions \mathcal{D} . (This is also common in the literature. E.g., [2] did so in the context of slow refresh.) Indeed, it is easy to see that our refresh procedure is too simple to work for arbitrary (even independent) distributions of entropy k . For example, if only k bits of X_0 have entropy, it is trivial to see where these k bits “travel” after i mixing steps given by π . Say, for rotation by 1, after i such rotations the first k bits $(1, \dots, k)$ go to locations $(1 + i \bmod n, \dots, k + i \bmod n)$. Thus, in this example (which is easy to generalize to any π) one can define X_i to be uniform over positions $(1 + i \bmod n, \dots, i + k \bmod n)$. This gives k independent bits of entropy, but this entropy is repeatedly added to the same place (just shifted over and over). Hence, we can never accumulate any entropy beyond the first k bits in this example.

Two-Monotone Distributions. Of course, the example above seems rather artificial, and unlikely to occur in the actual distributions encountered by these RNGs. (E.g., it seems implausible that the distribution of interrupt timings could have, say, the 10th bit uniformly random but the least significant bit fixed.) Thus, we must choose an appropriate family of distributions. We need *some* restriction on our sources to avoid the counterexamples above, but we would of course like to work with the most general class of distributions possible.

As our first main contribution, we provide a definition that is quite general but sufficient for our purposes. Indeed, as we will discuss more below, for this class of distributions, we are able to formalize the intuitive requirement that “natural distributions have most of their entropy in the lower-order bits.”

² See also [4, 5, 17] for some exciting advances in the area of randomness extraction from independent sources.

Specifically, we define a very wide class of distribution, which we call *2-monotone*. These are n -bit distributions such that the probability mass function over $\{0, \dots, 2^n - 1\}$ (i.e., interpreting the n bits as an integer written in binary) “has at most one peak.” (Formally, the distribution is 2-monotone if we can divide \mathbb{Z}_{2^n} into two intervals such that the probability mass function is monotone on the two intervals. See Sect. 3.) This is a large class, and it includes, e.g., Gaussians over \mathbb{Z}_{2^n} , exponential distributions over \mathbb{Z}_{2^n} , and uniform distributions over an interval—three natural distributions that one might use to model, e.g., the timing of interrupts.

We then show that any such distribution does in fact “have most of its entropy in the lower order bits.” (The precise statement is Fourier analytic. See Lemma 1.) This will help us overcome the impossibility result sketched above, while maintaining a (surprisingly!) large level of generality. To summarize, we will instantiate our family of distributions to be $\mathcal{D}_{k,n}$ —all two-monotone distributions on n bits having entropy at least k , and will allow arbitrary independent (but *not necessarily identical*) choices of entropic inputs $X_1, X_2, \dots \in \mathcal{D}_{k,n}$.

Goal: Entropy Accumulation. We must also select the notion of entropy for the register R for our goal of entropy accumulation. As our default choice, we will use the standard notion of min-entropy, $H_{\min}(R)$. This is a conservative notion of entropy which is enough to be composed with any Slow-Refresh procedure (or any other randomness extractor [20]) from the literature. However, some RNGs [7,9],³ and all randomness extractors based on the famous leftover hash lemma [15], can work for a less conservative notion of entropy, called *collision entropy* $H_2(R)$. Hence, in our results we will keep track of both the min- and the collision entropy of R .⁴ Indeed, our collision entropy results will be, as expected, slightly better than the min-entropy bounds.

Putting everything together, we arrive at the following clean question:

Main Question: *For given n, k , permutation π , and number of iterations ℓ , what is the min-/collision entropy of R_ℓ , where $R_0 = 0^n$, $R_i = A_\pi R_{i-1} \oplus X_i$, and X_1, X_2, \dots, X_ℓ are independent two-monotone distributions from $\mathcal{D}_{k,n}$?*

Bigger Picture. We stress once again that our question is largely complementary and incomparable to the analyses of “slow refresh” procedures from all the prior work [7,9,10,14,16]. Slow refresh operates on much larger block size $N \gg n$, is concerned with randomness extraction rather than accumulation, and attempts to defend against much more powerful attacks. In order to achieve this,

³ This is not stated in the results of [7,9], but is implicit from the technical analysis.

⁴ Our results will eventually give standard randomness extractors, when R accumulates nearly a full n bits of entropy because $\text{SD}(D, U) \leq \frac{1}{2} \cdot \sqrt{2^{n-H_2(D)} - 1}$, and $\text{SD}(D, U) \leq 2^{n-H_{\min}(D)} - 1$. However, we choose to focus on entropy accumulation, as (1) this is the use of superefficient entropy accumulators in real-world applications; (2) the restrictive format of our accumulators—while sufficient to quickly get to nearly n bits of entropy (which is our goal!)—will be wasteful in “squeezing the last few bits” of entropy needed for extraction.

the slow-refresh procedure must necessarily be much slower than our fast-refresh procedure. In particular, the two procedures are used in different, complementary places in the overall RNG design: the array or registers becomes an input to the slow-refresh procedure after many fast-refresh calls. In Sect. 8, we briefly discuss how our results might start filling the “missing link” in the prior RNG work, but stress once again that they cannot be meaningfully compared to each other.

1.2 Our Contributions

Rotation performs reasonably well. Recall that we show a key property of 2-monotone distributions: they “have most of its entropy in the lower order bits.” (The precise statement is Fourier analytic. See Lemma 1.) Using this characterization, we can then relatively easily show that any rotation on n bits (with α coprime to n , or, indeed, any cyclic permutation) can accumulate nearly a full n bits of entropy in n steps.

Theorem 1 (Informal, see Theorem 8). *Any rotation on n bits (with rotation number α coprime to n) will accumulate (approximately) $n(1 - 2^{-2k+2})$ bits of collision entropy and (approximately) $n(1 - 2^{-k+1})$ bits of min-entropy from any n independent sources in $\mathcal{D}_{k,n}$, for $k > 1$.*

Comparing different rotations using covering number. Theorem 1 justifies the use of rotation, but only if we are willing to wait n steps (regardless of how large k is) and fails to distinguish between different rotation numbers α . Indeed, as we discussed above, when $\alpha = 1$, we do in fact need roughly n steps in order to accumulate nearly n bits of entropy, even if the input already has very high entropy. So, if we wish to do better, we must somehow distinguish between different rotation numbers.⁵

To do this, we introduce a simple, efficiently computable quantity $C_{\alpha,k}$, which we call the *covering number*. Intuitively, $C_{\alpha,k}$ is the number of steps needed for rotation by α to accumulate full entropy when the input is uniform on $\{0, \dots, 2^k - 1\}$. Equivalently, $C_{\alpha,k}$ is the minimal number m such that $\{i + \alpha j \bmod n : 0 \leq i < k, 0 \leq j < m\} = [n]$, i.e., the minimal m such that “ m rotations of the first k bits are sufficient to cover all bits.” It is easy to see that the covering number is at least n/k and at most $n - k + 1$.

Notice that the covering number is exactly the number of steps needed to accumulate full entropy from the (two-monotone) distribution in which the first k bits are uniform and independent, while the remaining $n - k$ bits are fixed. We show (using Fourier-analytic techniques) that the covering number actually characterizes the performance of rotation by α on *all* 2-monotone distributions with entropy k , up to a factor of 2 in k . In other words, up to this factor of 2 in k (and ignoring the specific notion of “accumulating enough entropy”), the uniform distribution on $\{0, \dots, 2^k - 1\}$ is “the worst case”.

⁵ It is easy to see that all rotations perform identically if we wait exactly n steps. So, this question is essentially only interesting for fewer than n steps.

Theorem 2 (Informal, see Theorem 10). *Let $m := C_{\alpha, \lceil k/2 \rceil}$ and $k \geq 2$. After m steps, rotation by α accumulates at least $n \cdot (1 - 2^{-k})$ bits of collision entropy and $n \cdot (1 - 2^{-k/2})$ bits of min-entropy from any distribution in $\mathcal{D}_{k,n}$. Alternatively, it accumulates at least $n - 1$ bits of collision entropy after $m(1 + \log(n/k)/k)$ steps, and $n - 1$ bits of min-entropy after $m(1 + 2 \log(n/k)/k)$ steps.*

Theorem 2 suggests comparing rotations according to their covering numbers $C_{\alpha,k}$, effectively reducing a seemingly very difficult problem to a simple calculation. Therefore, we compute these covering numbers for different rotations. While there is no unambiguous ranking of the different rotations,⁶ we show that some rotations perform well in general, while others do not. (E.g., $C_{11,k} > n - 3k$ for $n = 32$.) In particular, Microsoft’s choice of $\alpha = 19$ when $n = 64$ is quite reasonable (though $\alpha \in \{15, 23, 27\}$ also seem like reasonable choices). Microsoft’s choice of $\alpha = 5$ for $n = 32$ is also reasonable, though we observe that certain other choices, particularly $\alpha = 7$ and $\alpha = 9$, also perform reasonably well for all k and perform noticeably better when the input has high entropy. See Figs. 3 and 4 for the data. (See [1] for a table with all covering numbers for $n = 32$ and $n = 64$.)

Other Permutations and Tightness. Our analysis of the covering number above extends immediately to any cyclic permutation $\pi : [n] \rightarrow [n]$. Specifically, the covering number $C_{\pi,k}$ of π essentially characterizes its behavior as an entropy accumulator when its input is a 2-monotone source with entropy k (up to a factor of 2 in k). In fact, in Theorem 11 we show that this generalization of Theorem 2 is quite tight. In particular, there exists a distribution $D \in \mathcal{D}_{k,n}$ (in fact, the same distribution that we use for our empirical results discussed below) such that no permutation π (including all rotations and the new permutation we discuss below) accumulates more than $n - 1$ bits of collision entropy from D in fewer than $n \log(n)/(k^2 + 4)$ steps. Similarly, it takes at least $2n \log(n)/(k^2 + 4)$ steps to accumulate $n - 1$ bits of min-entropy from this distribution.

Notice, in particular, that our upper and lower bounds nearly match when one sets $m \approx n/k$. (While $m = C_{\pi, \lceil k/2 \rceil}$ cannot be smaller than $2n/k$, as we describe below in more detail, we expect that this factor of two is an artifact of our proof and that taking $m \approx C_{\pi,k}$ is a good heuristic. Since $C_{\pi,k}$ can be as small as $\lceil n/k \rceil$, this suggests taking $m \approx n/k$.)

A different permutation: bit-reversed rotation. Our characterization of condensing in terms of $C_{\pi,k}$ motivates us to find a permutation π whose covering number $C_{\pi,k}$ is small for all k ; ideally, $C_{\pi,k} \approx n/k$, which is the minimal possible covering number. Indeed, in this regime, our condensing is provably the best possible: we accumulate almost all k bits of input entropy for each of the first nearly n/k steps.

⁶ Some rotations will perform very well for some k , and others will perform well for other k . E.g., for $\alpha = k$, $C_{\alpha,k} = \lceil n/k \rceil$ is always minimal. So every rotation has an optimal covering number for at least one choice of k .

To that end, we construct a permutation that we call *bit-reversed rotation*. This is the permutation obtained by (1) associating the i th bit with the $(\log_2 n + 1)$ -bit string i written in binary; (2) setting $\sigma(i)$ to be the number obtained by *reversing* this bit string; and (3) sending the i th bit to the unique position j with $\sigma(j) + 1 = \sigma(i) \bmod 2^n$. This permutation actually arises naturally from a simple greedy construction in this context,⁷ and it satisfies $C_{\pi,k} = n/k$ whenever n and k are both powers of two. I.e., it has optimal covering number $C_{\pi,k}$ for all powers of two k simultaneously! (For general k , the covering number is always bounded by $2n/k$; see Theorem 14.)

In Fig. 5, we compare covering numbers of bit-reversed rotation against covering numbers of rotation-by-5 for $n = 32$ and rotation-by-19 for $n = 64$ used by Microsoft (and the optimal value n/k). We see that bit-reversed rotation seems to perform at least as well as rotation, and better in several regions. Thus, while we leave it to practitioners to determine whether implementing our new permutation would be preferable in the context of their RNGs, our study suggests that it seems to be the most natural choice from a theoretical perspective. (More on this in our experimental results below.)

Experimental results to compute the exact number of samples needed.

Theorem 2 (and its generalization in Theorem 10) gives strong theoretical justification for using a cyclic permutation with low covering numbers. However, this loss of a factor of 2 in k (i.e., the fact that the theorem requires $C_{\pi, \lfloor k/2 \rfloor}$ samples instead of $C_{\pi,k}$ steps) is unfortunate—especially for the practical case that interests us most, in which n is a small constant like $n = 32$ or $n = 64$. For practical applications, we care about the fine-grained detail of the performance, and we expect that $C_{\pi,k}$ is in fact the right answer, as in the following heuristic.

We expect that (just slightly more than) $C_{\pi,k}$ steps should be sufficient to accumulate nearly full entropy from 2-monotone sources with entropy at least k .

This is of course true—essentially by definition—for the special case of the uniform distribution over $\{0, \dots, 2^k - 1\}$, and also gives us a lower bound for the general class $\mathcal{D}_{k,n}$.

To that end, in Theorem 16 we use the Fourier-analytic theoretical machinery that we used to prove Theorem 2 in order to derive a *closed form* expression for the exact (min- or collision) entropy accumulated by any permutation when the input source is an *exponential* distribution. (In other words, the distribution in which the probability that an interrupt happens at time t is proportional to $e^{-t/\sigma}$ for some $\sigma > 0$.) This is a natural example of a 2-monotone distribution (and far less trivial than the uniform distribution), and this closed form lets us compute exactly the number of samples needed to accumulate, say, $(n - 1)$ bits of min-/collision entropy.⁸

⁷ It also arises naturally in other contexts, such as in the fast Fourier transform (in the form of the bit-reversed involution, which we call σ above).

⁸ As we see from both our theoretical and our experimental results, our accumulators quickly collect almost n bits of entropy, at nearly optimal pace of k bits per sample,

These *exact* calculations allow us to answer three interesting questions, at least for the clean and natural case of exponential distributions:

1. Exactly how close is $C_{\pi,k}$ to the *actual* number of samples to accumulate nearly n (say, $(n - 1)$) bits of entropy close?
2. Does bit-reversed rotation perform at least as well as any rotation by α ?
3. How much faster does collision entropy accumulate compared to min-entropy?

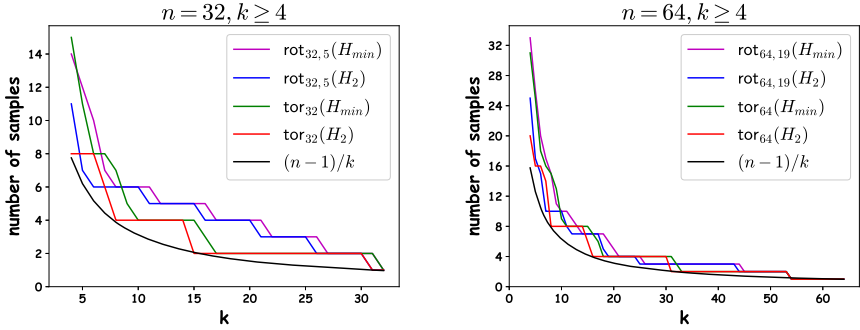


Fig. 1. Comparison between the exact number of samples needed to condense to 31 bits of collision/min-entropy (or 63 bits) from the exponential distribution, for bit-reversed rotation and the rotations used by Microsoft with input entropy k .

Our empirical results show that (at least for this natural distribution), (1) the true number of samples needed to accumulate from k bits of entropy to nearly n bits of entropy is very close to $C_{\pi,k}$; (2) bit-reversed rotation compares quite favorably with rotation by α ; and (3) collision entropy accumulates slightly but notably faster than min-entropy. There are some subtleties, though. (See Fig. 1 for the high-level picture.) For more detailed discussion, we refer readers to the full version of the paper [8].

Summary. Overall, we believe that our work provides both theoretical and practical results to shed light on a previously unexplored, but significant aspect of all practical RNGs: the design of “superefficient” entropy accumulation functions.

2 Preliminaries

For an integer $n \geq 1$, we write $[n] := \{0, \dots, n - 1\}$. For a distribution D over $\{0, 1\}^n$ and $\mathbf{x} \in \{0, 1\}^n$, we write $D(\mathbf{x}) := \Pr_{\mathbf{X} \sim D}[\mathbf{X} = \mathbf{x}]$ for the probability that D assigns to \mathbf{x} .

but squeezing the last couple of bits (i.e., becoming an extractor) takes many more samples. This is why we stop our experiments at $(n - 1)$ bits of entropy.

The *min-entropy* and *collision entropy* of D are

$$H_{\min}(D) := \min_{x \in \{0,1\}^n} \log_2(1/D(x)) \text{ and } H_2(D) := \log_2(1/\sum_x D(x)^2).$$

We will consider the problem of converting independent samples from a distribution D with some min-entropy into a new distribution with large min-/collision entropy.

For a distribution D over $\{0,1\}^n$ and $\mathbf{w} \in \{0,1\}^n$, we define the Fourier coefficient of D at \mathbf{w} as

$$\widehat{D}(\mathbf{w}) := \mathbb{E}_{\mathbf{X} \sim D} [(-1)^{\langle \mathbf{X}, \mathbf{w} \rangle}] = \Pr_{\mathbf{X} \sim D} [\langle \mathbf{X}, \mathbf{w} \rangle = 0 \pmod 2] - \Pr_{\mathbf{X} \sim D} [\langle \mathbf{X}, \mathbf{w} \rangle = 1 \pmod 2].$$

Fact 3. For any distribution D over $\{0,1\}^n$, and \mathbf{x} in $\{0,1\}^n$,

$$D(\mathbf{x}) = \frac{1}{2^n} \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}(\mathbf{w}) (-1)^{\langle \mathbf{x}, \mathbf{w} \rangle}.$$

Theorem 4 (Parseval’s theorem). For any distribution D over $\{0,1\}^n$,

$$\sum_{\mathbf{x} \in \{0,1\}^n} D(\mathbf{x})^2 = 2^{-n} \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}(\mathbf{w})^2.$$

Corollary 1. For any distribution D over $\{0,1\}^n$,

$$H_2(D) = n - \log_2 \left(\sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}(\mathbf{w})^2 \right),$$

and

$$H_{\min}(D) \geq n - \log_2 \left(\sum_{\mathbf{w} \in \{0,1\}^n} |\widehat{D}(\mathbf{w})| \right).$$

Corollary 1 shows that the sum of the squares of Fourier coefficients characterizes the collision entropy, and the sum of the absolute values of Fourier coefficients is useful for bounding min-entropy.

Proof. By Parseval’s theorem, we have

$$H_2(D) = \log_2(1/\sum_x D^2(x)) = \log_2(2^n / \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}^2(\mathbf{w})),$$

which implies the desired conclusion. By Fact 3,

$$\begin{aligned} H_{\min}(D) &= \min_{x \in \{0,1\}^n} \log_2(1/D(x)) \\ &= \min_{x \in \{0,1\}^n} \log_2(2^n / \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}(\mathbf{w}) (-1)^{\langle \mathbf{x}, \mathbf{w} \rangle}) \\ &\geq \log_2(2^n / \sum_{\mathbf{w} \in \{0,1\}^n} |\widehat{D}(\mathbf{w})|) \end{aligned}$$

as desired. □

The Fourier coefficients arise naturally in our context because they interact nicely with both convolution and linear transformations, as this next well-known claim shows.

Claim 5. For distributions D_1, \dots, D_m over $\{0, 1\}^n$ and linear transformations $A_1, \dots, A_m \in \mathbb{F}_2^{n \times n}$, let D be the distribution given by

$$\Pr_{\mathbf{X} \sim D} [\mathbf{X} = \mathbf{x}] = \Pr_{\mathbf{X}_1 \sim D_1, \dots, \mathbf{X}_m \sim D_m} [A_1 \mathbf{X}_1 \oplus \dots \oplus A_m \mathbf{X}_m = \mathbf{x}],$$

where the \mathbf{X}_i are independent. Then,

$$\widehat{D}(\mathbf{w}) = \widehat{D}_1(A_1^T \mathbf{w}) \cdots \widehat{D}_m(A_m^T \mathbf{w}).$$

for any $\mathbf{w} \in \{0, 1\}^n$.

Proof. We have

$$\begin{aligned} \mathbb{E}[(-1)^{\langle \mathbf{w}, \mathbf{X} \rangle}] &= \mathbb{E}[(-1)^{\langle \mathbf{w}, A_1 \mathbf{X}_1 \oplus \dots \oplus A_m \mathbf{X}_m \rangle}] \\ &= \mathbb{E}[(-1)^{\langle \mathbf{w}, A_1 \mathbf{X}_1 \rangle}] \cdots \mathbb{E}[(-1)^{\langle \mathbf{w}, A_m \mathbf{X}_m \rangle}] \\ &= \mathbb{E}[(-1)^{\langle A_1^T \mathbf{w}, \mathbf{X}_1 \rangle}] \cdots \mathbb{E}[(-1)^{\langle A_m^T \mathbf{w}, \mathbf{X}_m \rangle}] \\ &= \widehat{D}_1(A_1^T \mathbf{w}) \cdots \widehat{D}_m(A_m^T \mathbf{w}). \end{aligned}$$

□

For a distribution D over $\{0, 1\}^n$, integer $\ell \geq 1$, and linear transformation $A : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, we write $D_A^{(\ell)}$ for the distribution obtained by sampling $\mathbf{X}_1, \dots, \mathbf{X}_\ell$ independently and returning $\mathbf{X}_1 \oplus A \mathbf{X}_2 \oplus \dots \oplus A^{\ell-1} \mathbf{X}_\ell$.

3 Capturing Natural Distributions

In this section, we consider natural distributions over the integers (e.g., the kinds of distributions that one might expect for interrupt timings). We associate with each integer $0 \leq x < 2^n$ the vector $\mathbf{x} = (x_0, \dots, x_{n-1}) \in \{0, 1\}^n$ given by its binary representation. In other words, the $x_i \in \{0, 1\}$ are the unique bits that satisfy $x = \sum 2^i x_i$. For example, x might correspond to the timing of a keystroke.

We observe that many natural distributions are captured by the general class of 2-monotone distributions, which we define below. See Sect. 7 for examples of natural distributions that are 2-monotone.

Definition 1 (2-monotone distributions over \mathbb{Z}_{2^n}). A function $p : [2^n] \rightarrow [0, 1]$ is monotone over an interval $\{i_1, i_1 + 1, \dots, i_2\}$ if

$$p[i_1 \bmod 2^n] \leq \dots \leq p[i_2 \bmod 2^n] \text{ or } p[i_1 \bmod 2^n] \geq \dots \geq p[i_2 \bmod 2^n].$$

We say that p is 2-monotone over \mathbb{Z}_{2^n} , if there exist $0 \leq i_1 < i_2 \leq 2^n - 1$ such that p is monotone on the interval $\{i_1, \dots, i_2\}$ and on the interval $\{i_2, \dots, 2^n - 1, 2^n, \dots, 2^n + i_1 - 1\}$.

We say that a distribution D over $\{0, 1\}^n$ is 2-monotone over \mathbb{Z}_{2^n} if it is obtained by sampling an integer $0 \leq X \leq 2^n - 1$ (interpreted as a bit string as above) according to a 2-monotone probability mass function (Fig. 2).

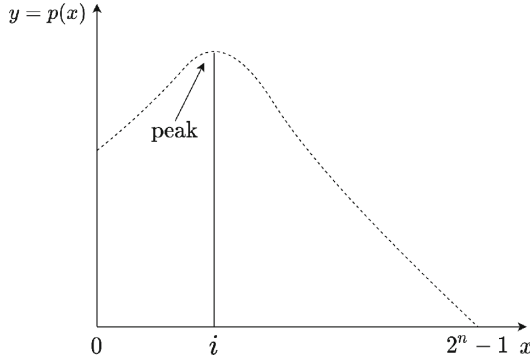


Fig. 2. A depiction of a 2-monotone distribution over \mathbb{Z}_{2^n}

Intuitively, 2-monotone distributions “change direction at most twice” when viewed as functions on the cycle \mathbb{Z}_{2^n} , so that they have “at most one peak” (and “at most one trough”). (For example, all unimodal distributions are 2-monotone.)

A very nice feature of 2-monotone distributions D is that $|\widehat{D}(\mathbf{w})|$ is small if $w_i = 1$ for some small index i . This formally captures the intuition that the lower-order bits of “natural distributions” should have high entropy.

Lemma 1. For any 2-monotone distribution D over $\{0, 1\}^n$ with min-entropy k , and $\mathbf{w} \in \{0, 1\}^n$ with $w_i = 1$,

$$|\widehat{D}(\mathbf{w})| \leq \min\{1, 2^{i+1-k}\}.$$

The lemma follows immediately from the following two claims.

Claim 6. If $\sum_{j=1}^{2^n-1} |D(j) - D(j - 1)| \leq \varepsilon$, then for any \mathbf{w} with $w_i = 1$,

$$|\widehat{D}(\mathbf{w})| \leq \min\{1, 2^i \cdot \varepsilon\}.$$

Proof. We have

$$\begin{aligned}
 |\widehat{D}(\mathbf{w})| &= \left| \mathbb{E}_{\mathbf{X} \sim D} [(-1)^{\langle \mathbf{X}, \mathbf{w} \rangle}] \right| \\
 &= \left| \sum_{\mathbf{X}: \mathbf{X}_i=0} (-1)^{\langle \mathbf{X}, \mathbf{w} \rangle} (D(\mathbf{X}) - D(\mathbf{X} + \mathbf{e}_i)) \right| \\
 &\leq \sum_{\mathbf{X}: \mathbf{X}_i=0} |D(\mathbf{X}) - D(\mathbf{X} + \mathbf{e}_i)| \\
 &= \sum_{\mathbf{X}: \mathbf{X}_i=0} |D(X) - D(X + 2^i)| \\
 &\leq \sum_{\mathbf{X}: \mathbf{X}_i=0} \sum_{j=1}^{2^i} |D(X + j) - D(X + j - 1)| \\
 &\leq 2^i \cdot \sum_{j=1}^{2^n-1} |D(j) - D(j - 1)|.
 \end{aligned}$$

□

Claim 7. *If D is 2-monotone over \mathbb{Z}_{2^n} with min-entropy k , then*

$$\sum_{j=1}^{2^n-1} |D(j) - D(j - 1)| \leq 2^{1-k}.$$

Proof. Suppose p is monotone on $\{0, \dots, i\}$ and $\{i, \dots, 2^n - 1\}$ for $0 < i < 2^n - 1$.

$$\begin{aligned}
 \sum_{j=1}^{2^n-1} |D(j) - D(j - 1)| &= \sum_{j=1}^i |D(j) - D(j - 1)| + \sum_{j=i}^{2^n-1} |D(j) - D(j - 1)| \\
 &= |D(i) - D(0)| + |D(2^n - 1) - D(i)| \\
 &\leq 2^{1-k}
 \end{aligned}$$

where the second inequality is by monotonicity of p , and the last inequality is because D has min-entropy k , so $|D(x) - D(y)| \leq 2^{-k}$ for every $0 \leq x, y \leq 2^n - 1$. Similarly, if p is monotone on $\{i_1, \dots, i_2\}$ and $\{i_2, \dots, 2^n - 1, 0, \dots, i_1\}$ for $0 < i_1 < i_2 < 2^n - 1$, we have

$$\sum_{j=1}^{2^n-1} |D(j) - D(j - 1)| \leq 2|D(i_1) - D(i_2)| \leq 2^{1-k}.$$

The desired conclusion follows.

□

4 Rotation Condensers

In this section, we set out to understand the power of rotation condensers generically. For integers $0 < \alpha < n$, we write $\text{rot}_{\alpha,n}$ for the linear transformation over $\{0, 1\}^n$ defined by

$$\text{rot}_{\alpha,n}((x_1, \dots, x_n)) := (x_{1+\alpha}, x_{2+\alpha}, \dots, x_n, x_1, \dots, x_\alpha) .$$

I.e., $\text{rot}_{\alpha,n}$ rotates the coordinates of a vector \mathbf{x} by α . Notice that $\text{rot}_{\alpha,n}^T = \text{rot}_{n-\alpha,n}$ and $\text{rot}_{\alpha,n}^k = \text{rot}_{k\alpha \bmod n,n}$.

Theorem 8. *For any $1 \leq \alpha < n$ with $\text{gcd}(\alpha, n) = 1$, and any 2-monotone distribution D over $\{0, 1\}^n$ with min-entropy $k > 1$, it holds that*

$$H_2(D_{\text{rot}_{\alpha,n}}^{(n)}) \geq n(1 - \log_2(1 + 2^{-2k+2})) \approx n(1 - 2^{-2k+2}) ,$$

$$H_{\min}(D_{\text{rot}_{\alpha,n}}^{(n)}) \geq n(1 - \log_2(1 + 2^{-k+1})) \approx n(1 - 2^{-k+1}) .$$

Theorem 8 provides some basic theoretical justification for the use of $\text{rot}_{\alpha,n}$ as a condenser. It shows rotation provably condenses to $\Omega(n)$ bits entropy within n steps.

Note that the theorem works for *any* rotation with $\text{gcd}(\alpha, n) = 1$, which means it also works for rotation whose rotation number is $\alpha = 1$. Although we typically think of rotation by 1 as the worst condenser (and we will show this in the next section), our result shows it can still condense 2-monotone distributions to linear entropy within n steps. What’s more, the proof of Theorem 8 immediately generalizes to *any* cyclic permutation,⁹ so that any cyclic permutation can condense to $\Omega(n)$ bits of entropy within n steps. (In particular, this can be applied to the cyclic permutation tor_n that we define in Sect. 6.)

Proof. Let $A := \text{rot}_{\alpha,n}$. For $\mathbf{w} \in \{0, 1\}^n$, we say that \mathbf{w} hits \mathbf{e}_0 if $w_0 = 1$, and say that $(\mathbf{w}, A^T \mathbf{w}, \dots, (A^T)^{n-1} \mathbf{w})$ hits \mathbf{e}_0 j times if there are j choices of i such that $(A^T)^i \mathbf{w}$ hits \mathbf{e}_0 . By Lemma 1, for every \mathbf{w} hitting \mathbf{e}_0 , it holds that $|\widehat{D}(\mathbf{w})| \leq 2^{-k+1}$.

Claim 9. *For $\mathbf{w} \in \{0, 1\}^n$, $(\mathbf{w}, A^T \mathbf{w}, \dots, (A^T)^{n-1} \mathbf{w})$ hits \mathbf{e}_0 exactly $|\mathbf{w}|$ times.*

Proof. It suffices to notice that for every i , there exists a distinct $0 \leq j < n$ such that $(A^T)^j \mathbf{e}_i = \mathbf{e}_0$ since $A = \text{rot}_{\alpha,n}$ and $\text{gcd}(\alpha, n) = 1$. For such an i and j , $(A^T)^j \mathbf{w}$ hits \mathbf{e}_0 if and only if $w_i = 1$. Therefore, the total number of hits is exactly the number of non-zero coordinates in \mathbf{w} . □

Given the claim, we note that

$$|\widehat{D_A^{(n)}}(\mathbf{w})| = \left| \prod_{i=0}^{n-1} \widehat{D}((A^T)^i \mathbf{w}) \right| \leq \prod_{i:(A^T)^i \mathbf{w} \text{ hits } \mathbf{e}_0} |\widehat{D}((A^T)^i \mathbf{w})| \leq (2^{-k+1})^{|\mathbf{w}|}$$

⁹ A cyclic permutation is a permutation $\sigma : [n] \rightarrow [n]$ such that for all $x \in [n]$, $\sigma^i(x) = x$ if and only if n divides i .

where the equality is by Claim 5, and the last inequality is by the claim and Lemma 1. Therefore,

$$\sum_{\mathbf{w} \in \{0,1\}^n} |\widehat{D_A^{(n)}}(\mathbf{w})|^2 \leq \sum_{\mathbf{w} \in \{0,1\}^n} (2^{-k+1})^{2|\mathbf{w}|} = \sum_{j=0}^n \binom{n}{j} (2^{-k+1})^{2j} = (1+2^{-2k+2})^n,$$

$$\sum_{\mathbf{w} \in \{0,1\}^n} |\widehat{D_A^{(n)}}(\mathbf{w})| \leq \sum_{\mathbf{w} \in \{0,1\}^n} (2^{-k+1})^{|\mathbf{w}|} = \sum_{j=0}^n \binom{n}{j} (2^{-k+1})^j = (1+2^{-k+1})^n.$$

Finally, by Corollary 1, we have

$$H_2(D_A^{(n)}) = n - \log\left(\sum_{\mathbf{w} \in \{0,1\}^n} |\widehat{D_A^{(n)}}(\mathbf{w})|^2\right) \geq n - \log(1+2^{-2k+2})^n$$

$$H_{\min}(D_A^{(n)}) \geq n - \log\left(\sum_{\mathbf{w} \in \{0,1\}^n} |\widehat{D_A^{(n)}}(\mathbf{w})|\right) \geq n - \log(1+2^{-k+1})^n$$

which imply the desired conclusion. □

5 Comparing Different Rotations and Permutations

In this section, we show how to compare the performance of different permutations on two-monotone distributions. For a permutation $\pi : [n] \rightarrow [n]$, we write A_π for the linear transformation over $\{0,1\}^n$ defined by

$$A_\pi(x_1, \dots, x_n) := (x_{\pi(1)}, \dots, x_{\pi(n)}).$$

I.e., A_π permutes coordinates of a vector \mathbf{x} by π . We slightly abuse notation and write $D_\pi^{(\ell)}$ to denote $D_{A_\pi}^{(\ell)}$.

We show that the rate of convergence of the condenser associated with a permutation π when run on two-monotone distributions with min-entropy k is governed by what we call the *covering number* $C_{\pi,k}$.

To get some intuition behind the covering number, consider the simple case when we want to extract from the distribution D that is uniform on $[2^k]$ (or, in terms of bit strings, uniform on $\{0,1\}^k \times \{0\}^{n-k}$). Notice that $D_\pi^{(m)}$ is the distribution that is uniform over the space spanned by $\{\mathbf{e}_i : \exists 0 \leq j < k, 0 \leq \ell < m, \pi^\ell(j) = i\}$.

In particular, the distribution $D_\pi^{(m)}$ has full entropy n if and only if $\{\pi^\ell(j) : 0 \leq j < k, 0 \leq \ell < m\}$ is the full set of coordinates $[n]$. We call the minimal such m the covering number of π , and the above discussion shows that it arises naturally in this context.

Definition 2 (Covering number). *For a permutation $\pi : [n] \rightarrow [n]$, and an integer $1 \leq k \leq n$, the covering number $C_{\pi,k}$ is the smallest natural number m such that*

$$\{\pi^\ell(j) : 0 \leq j < k, 0 \leq \ell < m\} = [n]$$

where $\pi^i = \pi \circ \pi^{i-1}$ for $i \geq 1$, π^0 is the identity function. (If no such m exists, we say $C_{\pi,k} = \infty$.)

To get some intuition for the covering number, first notice that we must have $C_{\pi,k} \geq \lceil n/k \rceil$. (This corresponds to the fact that we need at least $\lceil n/k \rceil$ sources with k bits of entropy each in order to have any hope of extracting n bits of entropy.) Of course, the covering number can be much worse than this, e.g., $C_{\text{rot}_{1,n},k} = n - k + 1$, which is the worst possible.

Also, notice that $C_{\text{rot}_{k,n},k} = \lceil n/k \rceil$. In other words, the optimal covering number $\lceil n/k \rceil$ is always achieved for fixed k by rotation by exactly k bits. (This suggests that, if your input is “nice enough,” e.g., 2-monotone, and you happen to *know* it has k bits of entropy, then rotating by $\alpha = k$ is a good idea. And, indeed, this is the case.) So, every choice of α has an optimal covering number for at least one choice of k , and we will not be able to unambiguously say that one choice of α is the “best”. Still, the performance of different choices of α over all $1 \leq k \leq n$ does vary considerably.

As we explained above, the covering number arises naturally when considering how quickly we can extract from the uniform distribution on $[2^k]$, which is perhaps the simplest 2-monotone distribution.

The following theorem shows that the covering number actually characterizes how quickly we converge to a high-entropy distribution for *any* 2-monotone distribution. In the proof, “we lose a factor of two in k ,” so that we need to take at least $C_{\pi,k/2}$ steps, rather than $C_{\pi,k}$ steps. And, we are of course only able to condense, not to extract. But, otherwise the result is tight. In Sect. 7.2, we show empirically that roughly $C_{\pi,k}$ steps is already enough for very strong condensing for natural distributions, suggesting that the factor of two loss is an artifact of the proof. Put together, the empirical data and the theoretical justification below strongly suggest that

The covering number $C_{\pi,k}$ is the right measure of how well a permutation π condenses for natural real-world distributions.

Theorem 10. *Let D be a two-monotone distribution with min-entropy at least k for some integer $k \geq 2$. Let $\pi : [n] \rightarrow [n]$ be a permutation with covering number $m := C_{\pi,k'}$, where $k' := \lfloor k/2 \rfloor$. Then for any $\ell \geq m$,*

$$H_2(D_\pi^{(\ell)}) \geq n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k' - k \lfloor \ell/m \rfloor}) \approx n(1 - 2^{k/2 - k\ell/m}),$$

$$H_{\min}(D_\pi^{(\ell)}) \geq n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k' - (k/2) \lfloor \ell/m \rfloor}) \approx n(1 - 2^{k/2 - k\ell/(2m)}).$$

Furthermore, there exists a two-monotone distribution D with min-entropy k such that for all $1 \leq \ell < C_{\pi,k}$

$$H_{\min}(D_\pi^{(\ell)}) = H_2(D_\pi^{(\ell)}) \leq n - (C_{\pi,k} - \ell).$$

In [8], we prove the following better lower bound on condensing (using the techniques developed in Sect. 7), showing that the results in Theorem 10 are quite tight. In fact, the distribution that we use to prove the theorem is exactly the same as the distribution that we use in our empirical results in Sect. 7.2. (This distribution arises naturally in this context.)

Theorem 11. *For every integer $1 \leq s \leq n$, there is a 2-monotone distribution D (in fact, a monotone distribution) with min-entropy $k > s - 1 + 1/2^s$ such that*

$$H_2(D_\pi^{(\ell)}) \leq n \cdot (1 - 2^{-s^2\ell/n - 4\ell/n}) \approx n(1 - 2^{-k^2\ell/n - 4\ell/n})$$

$$H_{\min}(D_\pi^{(\ell)}) \leq n \cdot (1 - 2^{-s^2\ell/(2n) - 2\ell/n}) \approx n(1 - 2^{-k^2\ell/(2n) - 2\ell/n}).$$

Notice how close these bounds are to the bounds in Theorem 10 with $m := n/k$.

5.1 Proof of Theorem 10

Proof. The “furthermore” part of the theorem is trivial. Indeed, it holds for the uniform distribution over $[2^k]$, which is clearly 2-monotone with min-entropy k . So, it remains only to prove the first statement.

Let $k' = \lfloor k/2 \rfloor$ and let B_0, B_1, \dots, B_{m-1} be a partition of $[n]$ such that $B_i \subseteq \pi^i([k'])$. Observe that $0 \leq |B_i| \leq k'$. We use \mathbf{w}_S to denote the projection of \mathbf{w} onto $S \subseteq [n]$. Let b be either 1 or 2.

Claim 12. $\sum_{\mathbf{w}} |\widehat{D}_\pi^{(\ell)}(\mathbf{w})|^b \leq \sum_{\mathbf{w}} |\widehat{D}_\pi^{(m)}(\mathbf{w})|^{b\lfloor \ell/m \rfloor}$.

Proof.

$$\begin{aligned} \sum_{\mathbf{w}} |\widehat{D}_\pi^{(\ell)}(\mathbf{w})|^b &= \sum_{\mathbf{w}} \prod_{i=0}^{\ell-1} |\widehat{D}((A_\pi^T)^i \mathbf{w})|^b \\ &\leq \sum_{\mathbf{w}} \prod_{j=0}^{\lfloor \ell/m \rfloor - 1} \prod_{i=0}^{m-1} |\widehat{D}((A_\pi^T)^{jm+i} \mathbf{w})|^b \\ &= \sum_{\mathbf{w}} \prod_{j=0}^{\lfloor \ell/m \rfloor - 1} |\widehat{D}_\pi^{(m)}((A_\pi^T)^{jm} \mathbf{w})|^b \\ &\leq \prod_{j=0}^{\lfloor \ell/m \rfloor - 1} \left(\sum_{\mathbf{w}} |\widehat{D}_\pi^{(m)}((A_\pi^T)^{jm} \mathbf{w})|^{b\lfloor \ell/m \rfloor} \right)^{1/\lfloor \ell/m \rfloor} \\ &= \prod_{j=0}^{\lfloor \ell/m \rfloor - 1} \left(\sum_{\mathbf{w}'} |\widehat{D}_\pi^{(m)}(\mathbf{w}')|^{b\lfloor \ell/m \rfloor} \right)^{1/\lfloor \ell/m \rfloor} \\ &= \sum_{\mathbf{w}} |\widehat{D}_\pi^{(m)}(\mathbf{w})|^{b\lfloor \ell/m \rfloor} \end{aligned}$$

where the first line is by Claim 5, the fourth line is by a claim proved in [8] and the fifth line is because $(A_\pi^T)^{jm}$ is a permutation over $\{0, 1\}^n$. □

Claim 13. $|\widehat{D}_\pi^{(m)}(\mathbf{w})|^{b\lfloor \ell/m \rfloor} \leq 2^{-a_w \cdot (bk/2)\lfloor \ell/m \rfloor}$, where $a_w := |\{i : \mathbf{w}_{B_i} \neq \mathbf{0}\}|$.

Proof. We say that \mathbf{w} hits $[k']$ if there exists an $i \in [k']$ such that $w_i = 1$. By Lemma 1, for any 2-monotone distribution with min-entropy at least k , and any \mathbf{w} which hits $[k']$, it holds that

$$|\widehat{D}(\mathbf{w})| \leq 2^{(k'-1)+1-k} \leq 2^{-k/2}.$$

For \mathbf{w} , let $S_w := \{i \in [m] : (A_\pi^T)^i \mathbf{w} \text{ hits } [k']\}$. Observe that, if $\mathbf{w}_{B_i} \neq \mathbf{0}$, because $B_i \subseteq \pi^i([k'])$, then

$$((A_\pi^T)^i \mathbf{w})_{[k']} = \mathbf{w}_{\pi^i([k'])} \neq \mathbf{0}$$

where $\pi^i([k']) := \{\pi^i(j) : j \in [k']\}$. I.e., $(A_\pi^T)^i \mathbf{w}$ hits $[k']$. Therefore, $|S_w| \geq a_w$. Moreover,

$$\begin{aligned} |\widehat{D}_\pi^{(m)}(\mathbf{w})|^{b\lfloor \ell/m \rfloor} &= \prod_{i=0}^{m-1} |\widehat{D}((A_\pi^T)^i(\mathbf{w}))|^{b\lfloor \ell/m \rfloor} \\ &\leq \prod_{i \in S_w} |\widehat{D}((A_\pi^T)^i \mathbf{w})|^{b\lfloor \ell/m \rfloor} \\ &\leq 2^{-|S_w| \cdot (bk/2)\lfloor \ell/m \rfloor} \\ &\leq 2^{-a_w \cdot (bk/2)\lfloor \ell/m \rfloor} \end{aligned}$$

as needed. □

We then prove Theorem 10 given above claims.

$$\begin{aligned} \sum_{\mathbf{w}} |\widehat{D}_\pi^{(\ell)}(\mathbf{w})|^b &\leq \sum_{\mathbf{w}} |\widehat{D}_\pi^{(m)}(\mathbf{w})|^{b\lfloor \ell/m \rfloor} \\ &\leq \sum_{\mathbf{w}} 2^{-a_w \cdot (bk/2)\lfloor \ell/m \rfloor} \\ &= \sum_{S \subseteq [m]} \binom{m}{|S|} \sum_{\mathbf{w} : \{i : \mathbf{w}_{B_i} \neq \mathbf{0}\} = S} 2^{-|S| \cdot (bk/2)\lfloor \ell/m \rfloor} \\ &= \sum_{S \subseteq [m]} \binom{m}{|S|} 2^{-|S| \cdot (bk/2)\lfloor \ell/m \rfloor} \prod_{i \in S} (2^{|B_i|} - 1) \\ &= \sum_{S \subseteq [m]} \binom{m}{|S|} \prod_{i \in S} ((2^{|B_i|} - 1) 2^{-(bk/2)\lfloor \ell/m \rfloor}) \\ &= \prod_{i=0}^{m-1} (1 + (2^{|B_i|} - 1) \cdot 2^{-(bk/2)\lfloor \ell/m \rfloor}) \end{aligned}$$

where the first inequality is by Claim 12, and the second inequality is by Claim 13. Furthermore, by a claim proved in [8], when $0 \leq |B_i| \leq k'$ and $\sum_{i=0}^{m-1} |B_i| = n$,

$$\prod_{i=0}^{m-1} (1 + (2^{|B_i|} - 1) \cdot 2^{-(bk/2)\lfloor \ell/m \rfloor}) \leq (1 + (2^{k'} - 1) \cdot 2^{-(bk/2)\lfloor \ell/m \rfloor})^{\lfloor n/k' \rfloor + 1}.$$

Finally, by Corollary 1, we have

$$H_2(D_\pi^{(\ell)}) \geq n - \log \left(\sum_w |\widehat{D_\pi^{(\ell)}}(\mathbf{w})|^2 \right) \geq n - (\lfloor \frac{n}{k'} \rfloor + 1) \cdot \log (1 + 2^{k' - k \lfloor \ell/m \rfloor}) ,$$

$$H_{\min}(D_\pi^{(\ell)}) \geq n - \log \left(\sum_w |\widehat{D_\pi^{(\ell)}}(\mathbf{w})| \right) \geq n - (\lfloor \frac{n}{k'} \rfloor + 1) \cdot \log (1 + 2^{k' - (k/2) \lfloor \ell/m \rfloor}) .$$

as needed. □

5.2 Covering Numbers of Different Rotations When $n = 32, 64$

In Figs. 3 and 4, we show the covering numbers of rotations with different rotation number α when $n = 32, 64$. In both cases, we compare the rotation numbers α chosen by Microsoft ($\alpha = 5$ for $n = 32$ and $\alpha = 19$ for $n = 64$) with other rotations that also perform well. (In Sect. 6, we show a different cyclic permutation, which is not a rotation, but has optimal covering number $C_{n,k} = n/k$ when both n and k are powers of two.) We also compare the covering numbers with the natural lower bound of $\lceil n/k \rceil$.

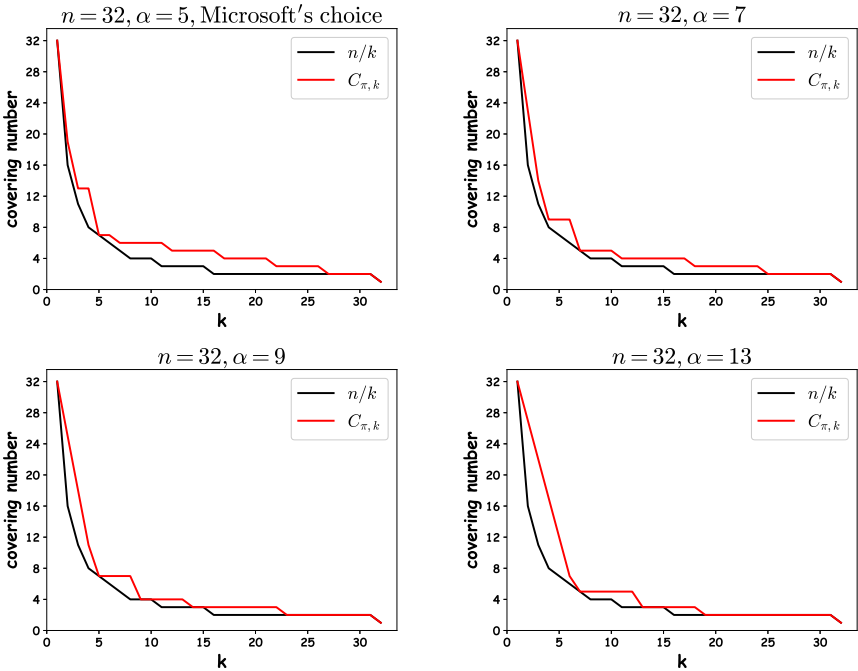


Fig. 3. Covering numbers of different rotations when $n = 32$

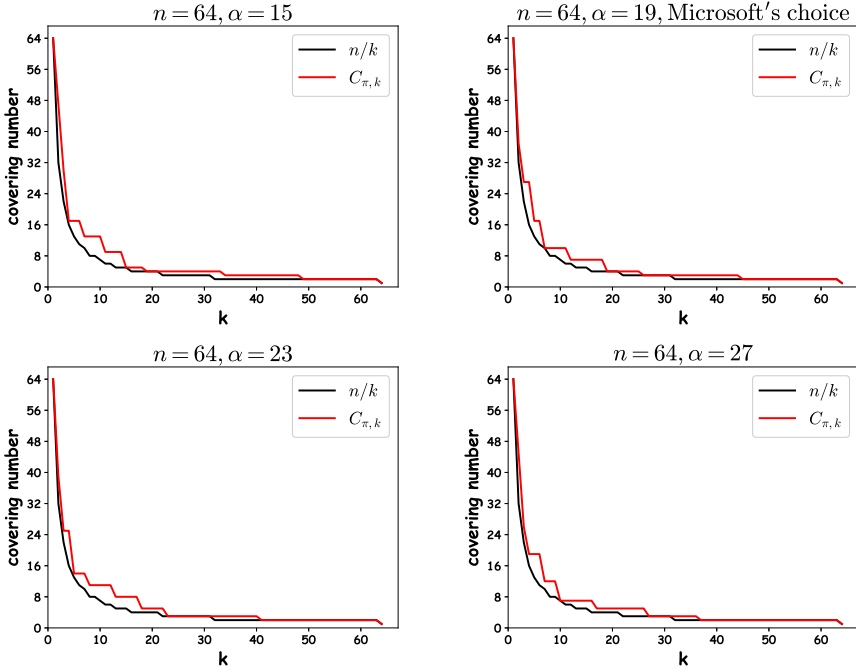


Fig. 4. Covering number of different rotations when $n = 64$

6 A New Recommendation: Bit-Reversed Rotation

Our characterization in terms of the covering number suggests the following “greedy” construction of a permutation with small covering number. Recall that we can write a cyclic permutation π in cycle notation as

$$a_0 := 0 \rightarrow a_1 := \pi(0) \rightarrow a_2 := \pi(\pi(0)) \rightarrow \dots \rightarrow a_{n-1} := \pi^{n-1}(0) \rightarrow a_n := 0.$$

So, suppose that n is a power of two, and suppose that we want to build some permutation $\pi : [n] \rightarrow [n]$ such that $C_{\pi,2} = n/2$ is as small as possible. Notice that this holds if and only if $a_{n/2} = 1$. I.e., “0 and 1 should be maximally far apart on the cycle”:

$$a_0 = 0 \rightarrow a_1 \rightarrow \dots \rightarrow a_{n/2-1} \rightarrow a_{n/2} = 1 \rightarrow a_{n/2+1} \rightarrow \dots \rightarrow a_{n-1} \rightarrow a_n = 0.$$

Similarly, $C_{\pi,4} = n/4$ if and only if $\{a_{n/4}, a_{n/2}, a_{3n/4}\} = \{1, 2, 3\}$, i.e., if and only if “0, 1, 2, and 3 are maximally far apart on the cycle” so that no two of them are within distance less than $n/4$. Therefore if we simultaneously have $C_{\pi,2} = n/2$ and $C_{\pi,4} = n/4$, then the permutation must have either the form

$$a_0 = 0 \rightarrow \dots \rightarrow a_{n/4} = 2 \rightarrow \dots \rightarrow a_{n/2} = 1 \rightarrow \dots \rightarrow a_{3n/4} = 3 \rightarrow \dots \rightarrow 0, \quad (1)$$

or

$$a_0 = 0 \rightarrow \dots \rightarrow a_{n/4} = 3 \rightarrow \dots \rightarrow a_{n/2} = 1 \rightarrow \dots \rightarrow a_{3n/4} = 2 \rightarrow \dots \rightarrow 0. \quad (2)$$

Continuing in this way, we see that we can build a cyclic permutation $\pi : [n] \rightarrow [n]$ such that $C_{\pi,2^a} = n/2^a$ for all integers $0 \leq a \leq \log_2 n$. In fact, we get a family of permutations (where the different members of the family vary as in Eqs. (1) and (2)), which represents all permutations that satisfy $C_{\pi,2^a} = n/2^a$ for all a . And, it is not hard to see that every such permutation has covering numbers given by $C_{\pi,k} = n/k'$, where $k' := 2^{\lfloor \log_2 k \rfloor}$ is the largest power of two smaller than k . (We prove this carefully below for one particular member of the family.)

Since all such permutations are essentially identical from our perspective, we choose one with a particularly elegant description. This elegant description might also help with efficient implementations. In particular, our choice, which we call bit-reversed rotation, is obtained by conjugating $\text{rot}_{1,n}$ with the well-studied and very efficient bit-reversal permutation.

Definition 3 (Bit-reversed rotation). For a power of two $n = 2^a$, the bit-reversal permutation $\sigma_n : [n] \rightarrow [n]$ is defined by

$$\sigma_n(b_0 + 2b_1 + \dots + 2^{a-1}b_{a-1}) = b_{a-1} + 2b_{a-2} + \dots + 2^{a-1}b_0$$

for $b_i \in \{0, 1\}$. (E.g., $\sigma_8(3) = 6$, and $\sigma_{16}(10) = 5$.) Notice that σ_n is an involution, i.e., $\sigma_n^{-1} = \sigma_n$. We nevertheless sometimes write σ_n^{-1} when this seems more natural.

Then, the bit-reversed rotation $\text{tor}_n : [n] \rightarrow [n]$ (tor is rot “reversed”) is given by $\text{tor}_n := \sigma_n^{-1} \circ \text{rot}_{1,n} \circ \sigma_n$. E.g., in cyclic notation, tor_8 is

$$0 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow 7 \rightarrow 0.$$

Equivalently, tor_n can be defined recursively via the recurrence $\text{tor}_{2n}(i + n) = \text{tor}_n(i)$ for $i < n$ together with the identity $\text{tor}_{2n}(i) = i + n$. (These two rules as simply describe binary addition, except with the bits reversed. I.e., “if the highest-order bit is 0, set it to 1; if it is 1, then set it to 0 and perform the same operation on the remaining bits.”)

Theorem 14. For a power of two n and $1 \leq k \leq n$,

$$C_{\text{tor}_n,k} = n/k' \leq 2n/k,$$

where $k' := 2^{\lfloor \log_2 k \rfloor}$ is the largest power of two that is no larger than k .

Proof. The result follows from the recurrence relation $C_{\text{tor}_{2n},k} = 2C_{\text{tor}_n,k}$ for $k \leq n$, together with two base cases, $C_{\text{tor}_n,n} = 1$ and $C_{\text{tor}_n,\ell} = 2$ for $n/2 \leq \ell < n$.

The first base case, $C_{\text{tor}_n,n} = 1$ is trivial. The second base case $C_{\text{tor}_n,\ell} = 2$ for $n/2 \leq \ell < n$ follows the simple observation that for all $i < n/2$, $\text{tor}_n(i) \geq n/2$.

This in particular implies that $C_{\text{tor}_n, n/2} = 2$, and since $1 < C_{\text{tor}_n, \ell} \leq C_{\text{tor}_n, n/2}$ for $n/2 \leq \ell < n$, we must have $C_{\text{tor}_n, \ell} = 2$ for all such ℓ .

To see the recurrence relation, notice that the recursive formula for tor_n implies that $\text{tor}_{2n}^2(i) = \text{tor}_n(i)$ for $i < n$. Applying this identity repeatedly gives $\text{tor}_{2n}^{2^\ell}(i) = \text{tor}_n^\ell(i)$. Similarly, $\text{tor}_{2n}^{2^{\ell+1}}(i) = \text{tor}_n^\ell(i) + n$. It follows that $i \in \{\text{tor}_n^\ell(0), \dots, \text{tor}_n^\ell(k-1)\}$ if and only if $i, i+n \in \{\text{tor}_{2n}^{2^\ell}(0), \dots, \text{tor}_{2n}^{2^\ell}(k-1), \text{tor}_{2n}^{2^{\ell+1}}(0), \dots, \text{tor}_{2n}^{2^{\ell+1}}(k-1)\}$, which immediately implies the recurrence relation.

Applying Theorem 10 immediately yields the following corollary, which shows that the bit-reversed rotation yields quite a good condenser. (In Sect. 7.2, we show empirical results that suggest even better performance, suggesting that the factor of 2 loss in k' is unnecessary.)

Corollary 2. *For any power of two n , and any 2-monotone distribution D with min-entropy at least $k \geq 2$, then for any $\ell \geq m$*

$$H_2(D_{\text{tor}_n}^{(\ell)}) \geq n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k'-k\lfloor \ell/m \rfloor}) \approx n \cdot (1 - 2^{-k^2 \ell / (2n)}) ,$$

$$H_{\min}(D_{\text{tor}_n}^{(\ell)}) \geq n - (\lfloor n/k' \rfloor + 1) \cdot \log_2(1 + 2^{k'-(k/2)\lfloor \ell/m \rfloor}) \approx n \cdot (1 - 2^{-k^2 \ell / (4n)}) ,$$

where $k' := \lfloor k/2 \rfloor$, and $m := n/2^{\lceil \log_2 k' \rceil}$ is the smallest power of two that is no smaller than n/k' .

In Fig. 5, we plot the covering numbers $C_{\text{tor}_n, k}$ together with $C_{\text{rot}_{\alpha, n}, k}$ for comparison.

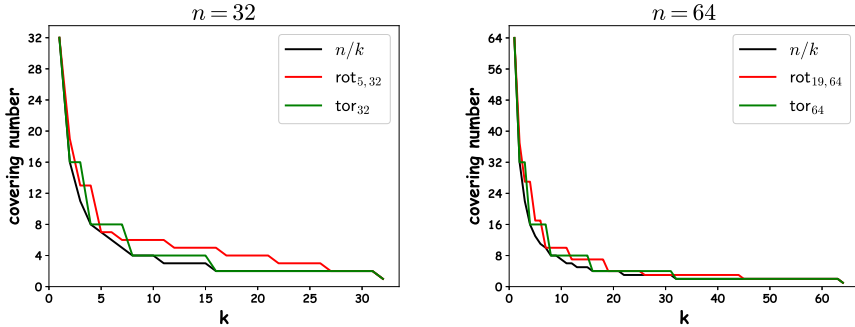


Fig. 5. The covering numbers of bit-reversed rotation and the two rotations $\text{rot}_{5,32}$ and $\text{rot}_{19,64}$ used by Microsoft. The n/k line represents the best possible value of $\lceil n/k \rceil$.

6.1 A Brief Note on Efficient Implementation

We leave it to practitioners to determine whether bit-reversed rotation can be implemented efficiently enough for their applications. However, we do note two things. First, we note that the bit-reversal permutation σ_n on which bit-reversed rotation is based is well-studied (in part because of its relationship with algorithms for the Fast Fourier Transform), with many fast implementations known (see, e.g., [18]).

Second, recall that we have defined our extractor according to the state update procedure $S_{i+1} \leftarrow \text{tor}_n(S_i) \oplus X = \sigma_n(\text{rot}_{1,n}(\sigma_n(S_i))) \oplus X$, where σ_n is the bit-reversal permutation. (Here, we have used the fact that $\sigma_n = \sigma_n^{-1}$, to replace $\sigma_n^{-1} \circ \text{rot}_{1,n} \circ \sigma_n$ with simply $\sigma_n \circ \text{rot}_{1,n} \circ \sigma_n$, since for implementation it seems quite useful to observe that these are the same map.) However, we note that one can equivalently use the rule $S'_{i+1} \leftarrow \text{rot}_{1,n}(S'_i) \oplus \sigma_n(X)$. I.e., one can simply perform the bit-reversal permutation on the input X and rotate the state S' by one. It is then easy to see that this rule maintains the invariant $S_i = \sigma_n(S'_i)$, and in particular, that S_i and S'_i have the same entropy. Therefore, one can use the second rule instead of the first, which could improve efficiency by replacing two applications of σ_n with a single application.¹⁰

7 Examples of Natural Distributions and Some Computational Results

Here, we list some natural distributions, all of which are 2-monotone. We then compute exactly the number of steps necessary to condense for the special case of the exponential distribution, which has a particularly nice form that makes such exact computation feasible.

Discrete Gaussian. For $s > 0$, we write $D_{\mathbb{Z},s}$ for the discrete Gaussian distribution over the integers with parameter s , i.e., defined by

$$\Pr_{X \sim D_{\mathbb{Z},s}} [X = z] = \frac{\exp(-\pi z^2/s^2)}{\sum_{z'=-\infty}^{\infty} \exp(-\pi (z')^2/s^2)}$$

for all integers $z \in \mathbb{Z}$.

Similarly, for $\sigma > 0$, the exponential distribution $E_{\mathbb{Z},\sigma}$ with parameter σ is the distribution over $\mathbb{Z}_{\geq 0}$ defined by

$$\Pr_{X \sim E_{\sigma}} [X = z] = \frac{\exp(-z/\sigma)}{\sum_{z'=0}^{\infty} \exp(-z'/\sigma)}$$

for all integers $z \geq 0$.

¹⁰ More generally, for any invertible linear transformations A, B , one can replace the rule $S_{i+1} \leftarrow B^{-1}AB(S_i) \oplus X$ with the equivalent rule $S'_{i+1} \leftarrow AS'_i \oplus BX$. This maintains the invariant $S_i = B^{-1}S'_i$.

Uniform distribution over an interval. For $0 \leq N_1 < N_2 \leq 2^n$, let U_{N_1, N_2} be the distribution over $\{0, 1\}^n$ obtained by sampling an integer $N_1 \leq X < N_2$ uniformly at random (interpreted as a bit string as above).

Shifted exponential distribution. For any $\sigma \geq 1$ and any integer $0 \leq N < 2^n$, let $E_{\sigma, N}$ be the distribution over $\{0, 1\}^n$ obtained by sampling an integer Y from an exponential distribution with parameter σ and setting $X := N + Y \bmod 2^n$ (and interpreting this as a bit string).

Shifted discrete Gaussian distribution. For any $s \geq 1$ and any integer $0 \leq N < 2^n$, let $D_{s, N}$ be the distribution over $\{0, 1\}^n$ obtained by sampling an integer Y from the discrete Gaussian distribution $D_{\mathbb{Z}, s}$ with parameter σ and setting $X := N + Y \bmod 2^n$ (and interpreting this as a bit string).

Claim 15. U_{N_1, N_2} , $E_{\sigma, N}$, and $D_{s, N}$ are all 2-monotone.

Proof. This fact is immediate for the uniform distribution U_{N_1, N_2} , by taking the monotone intervals $\{N_1, \dots, N_2 - 1\}$ and $\{N_2 - 1, \dots, 2^n + N_1 - 1\}$. Similarly, for $E_{\sigma, N}$, we can take the intervals $\{N - 1, N\}$ and $\{N, \dots, 2^n + N - 1\}$. For the Gaussian $D_{s, N}$, this follows from the fact that the function $t \mapsto \sum_{z \in \mathbb{Z}} e^{-\pi(z-t)^2}$ has maxima at $t \in \mathbb{Z}$, minima at $t \in \mathbb{Z} + 1/2$ and no other critical points, which one can verify, e.g., using the Poisson summation formula. \square

7.1 Entropy of Product Distributions Under Permutations

Below, we show an *exact* formula for the min-/collision entropy resulting from applying our permutation-based condensers to a product distribution. This exact formula is of course very useful, as it allows us to easily compute the min-/collision entropy of the state of our extractor, without directly computing the sum of 2^n Fourier coefficients. Indeed, in Sect. 7.2, we use this formula to show empirically that our extractor performs similarly as well with the unshifted exponential distribution—a product distribution.

Theorem 16. *Let D be a product distribution over $\{0, 1\}^n$ with $\Pr_{\mathbf{x} \sim D}[X_i = 0] = (1 + \varepsilon_i)/2$ for $\varepsilon_i \geq 0$. Then, for any cyclic permutation $\pi : [n] \rightarrow [n]$,*

$$H_2(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2 \left(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)} \right),$$

$$H_{\min}(D_\pi^{(\ell)}) = n - \sum_{i=0}^{n-1} \log_2 \left(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)} \right).$$

Proof. We have

$$\widehat{D_\pi^{(\ell)}}(\mathbf{w}) = \prod_{j=0}^{\ell-1} \widehat{D}(\pi^j(\mathbf{w})) = \prod_{j=0}^{\ell-1} \prod_{w_i=1} \widehat{D}(\mathbf{e}_{\pi^j(i)}) = \prod_{j=0}^{\ell-1} \prod_{w_i=1} \varepsilon_{\pi^j(i)} \geq 0$$

where the second equality is because D is a product distribution. Therefore,

$$D_\pi^{(\ell)}(\mathbf{x}) = \frac{1}{2^n} \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}_\pi^{(\ell)}(\mathbf{w})(-1)^{\langle \mathbf{x}, \mathbf{w} \rangle} \leq \frac{1}{2^n} \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}_\pi^{(\ell)}(\mathbf{w}) = D_\pi^{(\ell)}(0).$$

Moreover,

$$\begin{aligned} \sum_{\mathbf{w}} |\widehat{D}_\pi^{(\ell)}(\mathbf{w})|^2 &= \prod_{i=0}^{n-1} \left(1 + \prod_{j=0}^{\ell-1} |\widehat{D}(e_{\pi^j(i)})|^2 \right) = \prod_{i=0}^{n-1} \left(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)}^2 \right), \\ \sum_{\mathbf{w}} \widehat{D}_\pi^{(\ell)}(\mathbf{w}) &= \prod_{i=0}^{n-1} \left(1 + \prod_{j=0}^{\ell-1} \widehat{D}(e_{\pi^j(i)}) \right) = \prod_{i=0}^{n-1} \left(1 + \prod_{j=0}^{\ell-1} \varepsilon_{\pi^j(i)} \right). \end{aligned}$$

The equality for $H_2(D_\pi^{(\ell)})$ follows from Corollary 1, and the equality for $H_{\min}(D_\pi^{(\ell)})$ follows from

$$H_{\min}(D_\pi^{(\ell)}) = \log_2(1/D_\pi^{(\ell)}(0)) = \log_2(2^n / \sum_{\mathbf{w} \in \{0,1\}^n} \widehat{D}_\pi^{(\ell)}(\mathbf{w})).$$

□

Corollary 3. For any $\sigma \geq 1$, let $D := E_\sigma$ be the distribution over $\{0,1\}^n$ obtained by sampling an integer Y from an exponential distribution with parameter σ and setting $X := Y \bmod 2^n$ (and interpreting this as a bit string), as described above. Then, for any cyclic permutation $\pi : [n] \rightarrow [n]$,

$$\begin{aligned} H_2(D_\pi^{(\ell)}) &= n - \sum_{i=0}^{n-1} \log_2 \left(1 + \left(\prod_{j=0}^{\ell-1} \frac{1 - \exp(-2^{\pi^j(i)}/\sigma)}{1 + \exp(-2^{\pi^j(i)}/\sigma)} \right)^2 \right) \\ H_{\min}(D_\pi^{(\ell)}) &= n - \sum_{i=0}^{n-1} \log_2 \left(1 + \prod_{j=0}^{\ell-1} \frac{1 - \exp(-2^{\pi^j(i)}/\sigma)}{1 + \exp(-2^{\pi^j(i)}/\sigma)} \right). \end{aligned}$$

Proof. For any $0 \leq x < 2^n$,

$$\Pr_{X \sim D}[X = x] = \frac{\sum_{z \geq 0} \exp(-(x + 2^n z)/\sigma)}{\sum_{z \geq 0} \exp(-z/\sigma)} = C_\sigma \cdot \prod_i \exp(-2^i x_i / \sigma),$$

where

$$C_\sigma := \frac{\sum_{z \geq 0} \exp(-2^n z / \sigma)}{\sum_{z \geq 0} \exp(-z / \sigma)}.$$

I.e., D is a product distribution. From the above expression, we see that

$$\Pr_{X \sim D}[X_i = 0] = \exp(2^i / \sigma) \cdot \Pr_{X \sim D}[X_i = 1].$$

The desired conclusion then follows from Theorem 16 with

$$\varepsilon_i = \frac{1 - \exp(-2^i / \sigma)}{1 + \exp(-2^i / \sigma)}.$$

□

7.2 Computational Results for $n = 32$ and $n = 64$

Finally, we use the formula from Corollary 3 to directly compute the number of samples needed to condense to nearly full entropy from the exponential distribution with different starting entropy and different permutations. At a high level, these results confirm that the covering number $C_{\pi,k}$ provides a good estimate for the number of steps needed to condense. We display a more detailed discussion in the full version [8].

8 Bigger Picture

In this work we abstracted out and analysed the fast entropy accumulation procedures found in modern RNGs. We can now combine our results with prior RNG literature [7,9,10,14,16] to get a better big picture guarantee of the resulting RNG, but we start with some observations.

First, every RNG so far used a different (and often incompatible) modeling of the security of the slow refresh procedure. So, it’s not clear what is the “right” model for slow refresh—let alone a hybrid fast/slow model. The good news is that all of the slow refresh models share one thing in common—their rate of convergence depends only on either the overall collision/min-entropy that they received. So, the fact that our work guarantees entropic output from a fast refresh seems like a good start in trying to unify the two models.

Indeed, our results *do* (trivially) combine with every prior RNG work, and for concreteness we state one such combination below (focusing on the slow refresh RNG work of [10], but other combinations are done analogously). The main issue with such naive combinations is that they appear to only work with a relatively weak RNG adversary, which must output independent (but not necessarily identical) samples from the family of two-monotone distributions $\mathcal{D}_{k,n}$, for (fixed but unknown) min-entropy k per sample. We show a concrete example of a combined slow-refresh and fast-refresh procedure in the full version [8], but we stress that this is meant only as a proof of concept and that we do not claim that the model used in [8] is the “right” model.

As a positive, this is already a highly non-trivial and quite interesting model. For example, very related “constant entropy rate” adversaries were mentioned by Fergusson and Schneier [13]—and later formalized by [10]—in their design and analysis of Fortuna, which directly led to the Windows 10 RNG [12]. On the negative side, prior work on slow refresh [9,10,14,16]

[7] worked hard to give a lot of power to the RNG attacker, including the ability to output correlated samples, and drastically change the entropy of each sample (subject only to providing enough entropy overall). Thus, it is unfortunate that the naive composition that follows from using our work did not use all these powerful security guarantees of the slow-refresh procedures, and resulted in a much weaker overall attacker.

We note that, while the naive composition currently does not capture the ability of the distribution sampler to change its entropy parameter k , it is clear that the final RNG is at least somewhat resilient and easily adaptable to this

change, as the RNG design does not use the knowledge of k , and simultaneously provides good guarantees for all k . So it feels the actual hybrid slow-/fast-refresh RNG is much more robust than the current composition states. If nothing else, the resulting RNGs are basically what is used in the real world, and these RNGs appear to work well against practical entropy sources. In particular, while it is great that the standards for the slow refresh procedure in the literature are very high, the existing entropy sources (e.g., modeling timing of interrupts) appear to be much less adversarial, and likely lie somewhere in between the independent 2-monotone sources modeled in this paper and the very general classes handled in the literature on slow refresh procedures.

In summary, we view our current work as only the starting point in trying to understand and model the overall security of the composed RNG, and believe that finding the “right” model to combine fast refresh with slow refresh is a great avenue for future work.

Acknowledgments. Yevgeniy Dodis—Partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

Siyao Guo—Supported by Shanghai Eastern Young Scholar Program SMEC-0920000169. Parts of this work were done while visiting the Centre for Quantum Technologies, National University of Singapore.

Noah Stephens-Davidowitz—Some of this work was done at MIT supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS1718161, Microsoft Faculty Fellowship and an MIT/IBM grant. Some of this work was done at the Simons Institute in Berkeley.

References

1. Supplementary material: covering numbers for all rotations with $n = 32$ and $n = 64$. http://noahsd.com/rotation_extractor_supplement.zip
2. Bar-Yossef, Z., Trevisan, L., Reingold, O., Shaltiel, R.: Streaming computation of combinatorial objects. In: Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, 21–24 May 2002, pp. 165–174. IEEE Computer Society (2002). <https://doi.org/10.1109/CCC.2002.1004352>
3. Barak, B., Halevi, S.: A model and architecture for pseudo-random generation with applications to `/dev/random`. In: Atluri, V., Meadows, C.A., Juels, A. (eds.) Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, 7–11 November 2005, pp. 203–212. ACM (2005). <https://doi.org/10.1145/1102120.1102148>
4. Barak, B., Impagliazzo, R., Wigderson, A.: Extracting randomness using few independent sources. In: 45th Symposium on Foundations of Computer Science (FOCS 2004), Rome, Italy, 17–19 October 2004, Proceedings, pp. 384–393. IEEE Computer Society (2004). <https://doi.org/10.1109/FOCS.2004.29>
5. Chattopadhyay, E., Zuckerman, D.: Explicit two-source extractors and resilient functions. In: Wichs, D., Mansour, Y. (eds.) Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, 18–21 June 2016, pp. 670–683. ACM (2016). <https://doi.org/10.1145/2897518.2897528>

6. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.* **17**(2), 230–261 (1988). <https://doi.org/10.1137/0217015>
7. Coretti, S., Dodis, Y., Karthikeyan, H., Tessaro, S.: Seedless fruit is the sweetest: random number generation, revisited. In: Boldyreva, A., Micciancio, D. (eds.) *CRYPTO 2019, Part I*. LNCS, vol. 11692, pp. 205–234. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_8
8. Dodis, Y., Guo, S., Stephens-Davidowitz, N., Xie, Z.: No time to hash: on super-efficient entropy accumulation (full version). *Cryptology ePrint Archive*, Report 2021/523 (2021). <https://eprint.iacr.org/2021/523>
9. Dodis, Y., Pointcheval, D., Ruhault, S., Vergnaud, D., Wichs, D.: Security analysis of pseudo-random number generators with input: /dev/random is not robust. In: *CCS* (2013). <https://doi.org/10.1145/2508859.2516653>
10. Dodis, Y., Shamir, A., Stephens-Davidowitz, N., Wichs, D.: How to eat your entropy and have it too: optimal recovery strategies for compromised RNGs. *Algorithmica* **79**(4), 1196–1232 (2017). <https://doi.org/10.1007/s00453-016-0239-3>
11. Ferguson, N.: Private communication (2013)
12. Ferguson, N.: The windows 10 random number generation infrastructure (2019). <https://www.microsoft.com/security/blog/2019/11/25/going-in-depth-on-the-windows-10-random-number-generation-infrastructure/>. Accessed October 2019
13. Ferguson, N., Schneier, B.: *Practical Cryptography*. Wiley, New York (2003)
14. Gazi, P., Tessaro, S.: Provably robust sponge-based PRNGs and KDFs. In: Fischlin, M., Coron, J.-S. (eds.) *EUROCRYPT 2016*. LNCS, vol. 9665, pp. 87–116. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_4
15. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999). <https://doi.org/10.1137/S0097539793244708>
16. Hutchinson, D.: A robust and sponge-like PRNG with improved efficiency. In: Avanzi, R., Heys, H. (eds.) *SAC 2016*. LNCS, vol. 10532, pp. 381–398. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_21
17. Kamp, J., Rao, A., Vadhan, S.P., Zuckerman, D.: Deterministic extractors for small-space sources. *J. Comput. Syst. Sci.* **77**(1), 191–220 (2011). <https://doi.org/10.1016/j.jcss.2010.06.014>
18. Karp, A.H.: Bit reversal on uniprocessors. *SIAM Rev.* **38**(1), 1–26 (1996). <https://doi.org/10.1137/1038001>
19. Kelsey, J., Schneier, B., Ferguson, N.: Yarrow-160: notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In: Heys, H., Adams, C. (eds.) *SAC 1999*. LNCS, vol. 1758, pp. 13–33. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46513-8_2
20. Nisan, N., Zuckerman, D.: Randomness is linear in space. *J. Comput. Syst. Sci.* **52**(1), 43–52 (1996). <https://doi.org/10.1006/jcss.1996.0004>
21. Raz, R., Reingold, O.: On recycling the randomness of states in space bounded computation. In: Vitter, J.S., Larmore, L.L., Leighton, F.T. (eds.) *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, Atlanta, Georgia, USA, 1–4 May 1999, pp. 159–168. ACM (1999). <https://doi.org/10.1145/301250.301294>
22. Reingold, O., Shaltiel, R., Wigderson, A.: Extracting randomness via repeated condensing. In: *41st Annual Symposium on Foundations of Computer Science, FOCS 2000*, Redondo Beach, California, USA, 12–14 November 2000, pp. 22–31. IEEE Computer Society (2000). <https://doi.org/10.1109/SFCS.2000.892008>
23. Wikipedia: /dev/random (2004). <http://en.wikipedia.org/wiki/dev/random>. Accessed 09 Feb 2014

Protocols



A Logarithmic Lower Bound for Oblivious RAM (for All Parameters)

Ilan Komargodski^{1,2}(✉) and Wei-Kai Lin³

¹ Hebrew University, Jerusalem, Israel
ilank@cs.huji.ac.il

² NTT Research, Sunnyvale, CA, USA

³ Cornell University, Ithaca, USA
wklin@cs.cornell.edu

Abstract. An Oblivious RAM (ORAM), introduced by Goldreich and Ostrovsky (J. ACM 1996), is a (probabilistic) RAM that hides its access pattern, i.e., for every input the observed locations accessed are similarly distributed. In recent years there has been great progress both in terms of upper bounds as well as in terms of lower bounds, essentially pinning down the smallest overhead possible in various settings of parameters.

We observe that there is a very natural setting of parameters in which *no* non-trivial lower bound is known, even not ones in restricted models of computation (like the so called balls and bins model). Let N and w be the number of cells and bit-size of cells, respectively, in the RAM that we wish to simulate obliviously. Denote by b the cell bit-size of the ORAM. All previous ORAM lower bounds have a multiplicative w/b factor which makes them trivial in many settings of parameters of interest.

In this work, we prove a new ORAM lower bound that captures this setting (and in all other settings it is at least as good as previous ones, quantitatively). We show that any ORAM must make (amortized)

$$\Omega\left(\log\left(\frac{Nw}{m}\right) / \log\left(\frac{b}{w}\right)\right)$$

memory probes for every logical operation. Here, m denotes the bit-size of the local storage of the ORAM. Our lower bound implies that logarithmic overhead in accesses is necessary, even if $b \gg w$. Our lower bound is tight for *all* settings of parameters, up to the $\log(b/w)$ factor. Our bound also extends to the non-colluding multi-server setting.

As an application, we derive the first (unconditional) separation between the overhead needed for ORAMs in the *online* vs. *offline* models. Specifically, we show that when $w = \log N$ and $b, m \in \text{poly} \log N$, there exists an offline ORAM that makes (on average) $o(1)$ memory probes per logical operation while every online one must make $\Omega(\log N / \log \log N)$ memory probes per logical operation. No such previous separation was known for any setting of parameters, not even in the balls and bins model.

Keywords: Oblivious RAM · Lower bound · Cell-probe model

The full version is posted on Cryptology ePrint Archive, Report 2020/1132 [29].

© International Association for Cryptologic Research 2021

T. Malkin and C. Peikert (Eds.): CRYPTO 2021, LNCS 12828, pp. 579–609, 2021.

https://doi.org/10.1007/978-3-030-84259-8_20

1 Introduction

An oblivious RAM (ORAM), introduced by Goldreich and Ostrovsky [22], is a probabilistic RAM machine whose goal is to simulate an arbitrary RAM program while ensuring observable access patterns do not reveal information neither about the underlying data nor about the program being executed. This is obtained by making sure that any two sequences of *logical* operations on the memory (either reads or writes) translate into *indistinguishable* sequences of physical probes to the memory. ORAMs have become an indispensable tool in the design of cryptographic systems where it is necessary to make the observable access pattern independent of the underlying sensitive data. Somewhat surprisingly, this task comes up not only in the context of software protection, as originally suggested by [22], but also in less directly related contexts such as the design of secure processor [15, 16], secure multi-party computation [5, 21, 25, 38, 40, 55], and other central notions in computer science [4, 6, 9, 12, 20, 37, 39, 46, 51, 52, 58, 60].

A trivial way to construct an ORAM is to replace every logical access with a scan of the entire memory. While this solution is perfectly secure, it is highly inefficient and so the question is how efficient could an ORAM be compared to an insecure RAM. The primary efficiency metric of interest is:

I/O efficiency: The total number of physical probes to the memory of the ORAM amortized per logical operation.

Some previous works use bandwidth as the metric, but we chose to use I/O efficiency as our central metric since it is robust and well-defined in various ORAM settings. I/O efficiency can be translated into communication/bandwidth by multiplying by the ORAM cell size. See Remark 2.

Following Boyle and Naor [7], we shall distinguish between two classes of ORAM schemes: *offline* and *online*. An ORAM scheme is online if it supports accesses arriving in an online manner, one by one. An ORAM scheme is offline if it requires all accesses to be specified at once in advance. Most known ORAM constructions (e.g., [3, 10, 22, 24, 30, 41, 48, 50, 54]) work in the online setting as well with few exceptions (e.g., [7, 28, 47]). Also, most applications of ORAM schemes require that the scheme is online.

Existing lower bounds. Assume that the goal is to obviously simulate a RAM of N cells each of size w bits on a RAM with N' cells each of size b bits and using a local storage of size m bits. In the original work of Goldreich and Ostrovsky [22] it was shown that any ORAM scheme (even offline ones) must have I/O efficiency^{1,2}

$$\Omega\left(\frac{w}{b} \cdot \frac{\log N}{1 + \log(m/b)}\right).$$

¹ To the best of our knowledge, the lower bound technique of [22] was never analyzed without assuming that $b = w$. For completeness, we add a proof in the full version [29]. The bound that we state here is a little bit simplified for presentation purposes.

² Throughout this paper, unless otherwise stated, \log stands for \log_2 .

In one sense, this lower bound is very powerful: (1) It is pretty robust to the choice of w and b as long as $b = w$, (2) it can be cast for few other efficiency metrics besides I/O (see [54] for details), and (3) it applies to schemes that have $O(1)$ statistical failure probability. However, as observed by Boyle and Naor [7] this lower bound only applies to schemes in the so called “balls and bins” model³ which do not use cryptographic assumptions, leaving the possibility of more efficient constructions outside of this model.

In a beautiful recent work, Larsen and Nielsen [33, Theorem 2] proved a lower bound that applies to any online ORAM scheme, even ones that are not in the balls and bins model and ones that use cryptographic assumptions. They prove that any online ORAM must have I/O efficiency

$$\Omega\left(\frac{w}{b} \cdot \log\left(\frac{Nw}{m}\right)\right).$$

Similarly to the lower bound of Goldreich and Ostrovsky [22], this lower bound is also pretty robust to the choice of b and w as long as $b = w$.

Is sub-logarithmic efficiency possible? The above two lower bounds become completely trivial in the setting where, say, $w = \log N$ and $b, m \in \Theta(\log^2 N)$. In this case, both lower bounds simplify to $\Omega(1)$. This is by no means an esoteric setting of parameters. It is quite common and natural to consider RAM algorithms that take advantage of being able to place multiple elements in one cell and process all of them within a single memory access. Indeed, there is a long line of work in core algorithms literature designing efficient algorithms and studying tradeoffs in this setting (e.g., [2, 17, 23, 53]).

Focusing on oblivious sorting, one notable result is due to Goodrich [23] (see also a follow-up by Chan et al. [11]⁴) who showed an oblivious sorting algorithm that sorts N elements each of size w bits with $O((Nw/b) \cdot \log_{m/b}(Nw/b))$ memory probes on a RAM with cells of size b bits and local storage of size m bits. Setting $w = \log N$ and $b, m \in O(\log^3 N)$ (see also the full version [29] for the parameterization), we obtain an oblivious sorting algorithm with $O(N)$ memory probes. In contrast, when $w = b$ we have existing $\Omega(N \cdot \log N)$ lower bounds on the number of memory probes, either in the balls and bins model [36] or assuming a well-known network coding conjecture [14].

Oblivious sorting is one of the core building blocks in the design of many oblivious RAM constructions (for example, [3, 10, 22, 24, 30, 41]), suggesting that it may be possible to use the algorithms of [11, 23] to get an ORAM construction with sub-logarithmic I/O efficiency. This direction was pursued first by Goodrich and Mitzenmacher [23, 24] and then by Chan et al. [11], but they were only able

³ In the balls and bins model, items are modeled as “balls”, CPU registers and server-side data storage locations are modeled as “bins”, and the set of allowed data operations consists *only* of moving balls between bins. See the full version [29] for the definition of the model.

⁴ Chan et al. [11]’s algorithm has the same asymptotic efficiency and it is additionally in the balls and bins model.

to construct an ORAM with $O(\log N)$ I/O efficiency,⁵ assuming that $w = \log N$ and $b, m \in O(\log^3 N)$. By now, we already have an ORAM construction, due to Asharov et al. [3], with $O(\log N)$ I/O efficiency assuming only $w = b$ and $m \in O(b)$.

Given the state of affairs, it is an intriguing question whether more efficient ORAM constructions exist when $b \gg w$:

Is the linear dependence on w/b necessary? Alternatively, is it possible to break the logarithmic barrier for ORAM efficiency if $b \gg w$?

1.1 Our Results

In this work, we answer the above question *negatively* by showing that any online ORAM construction, including ones that are not in the balls and bins model and perhaps use cryptographic assumptions, cannot go below the logarithmic I/O efficiency barrier even if $b \gg w$. Restricted to online schemes, for a wide ranges of parameters, our lower bound improves on the lower bound of Goldreich and Ostrovsky [22] as well as the one of Larsen and Nielsen [33]. Specifically, we prove the following theorem.

Theorem 1 (Informal; See Theorem 3). *Consider a RAM with memory of N cells, each of size w bits. Any online ORAM that simulates such a RAM using cells of size b bits and local storage of size m bits, must have I/O efficiency*

$$\Omega\left(\log\left(\frac{Nw}{m}\right) / \left(1 + \log\left(\frac{b}{w}\right)\right)\right).$$

When $b = w$, our lower bound is identical to the one of Larsen and Nielsen [33] and is at least as good as the one of Goldreich and Ostrovsky [22]. However, when $b \in \omega(w)$, our lower bound is already better than both. For example, when $w = \log N$ and $b, m \in O(\log^c N)$ for any $c \geq 2$, our lower bound is $\Omega(\log N / \log \log N)$ while the ones of Goldreich and Ostrovsky [22] and Larsen and Nielsen [33] are both only $\Omega(1)$. As in [33]’s lower bound, our lower bound applies to ORAM schemes satisfying computational indistinguishability only with probability p and having δ failure probability in correctness for some fixed constants $0 < p, \delta < 1$. While this makes schemes somewhat weak, this only makes our lower bound stronger. Lastly, let us mention that our technique is pretty general and can be used to extend and improve other related lower bounds when $b \gg w$ (see Sect. 1.2 for pointers). For example, in the full version [29] we extend our lower bound to apply to the non-colluding multi-server setting, improving the recent lower bound of Larsen et al. [34] whenever $b \gg w$.

⁵ Actually, these works [11, 24] give ORAM constructions in a more general model called the *external memory* model, where there are three entities, a CPU, a cache, and a memory. The standard ORAM setting (which we consider here) is a special case of that model.

We remark that our lower bound in Theorem 1 is tight for all settings of parameters up to the $\log(\mathbf{b}/\mathbf{w})$ factor. This is due to the construction of Asharov et al. [3] who constructed an ORAM with $O(\log N)$ I/O efficiency for all values of $\mathbf{w} \geq \log N$ assuming only $m \geq \mathbf{b} \geq \mathbf{w}$ (and assuming that one-way functions exist).⁶

Separating offline and online ORAM. We use Theorem 1 to obtain the first separation between offline and online ORAM schemes. Specifically, we show that when we want to obliviously simulate a RAM with N cells of logarithmic size using a RAM with cells and local storage of poly-logarithmic size (in N), then there is an offline ORAM with $o(1)$ I/O efficiency while every online ORAM must have $\tilde{\Omega}(\log N)$ I/O efficiency. This separation is essentially optimal in terms of the gap between the cost of the offline and the online oblivious simulations.

Theorem 2 (Informal; See Theorem 6). *Consider the task of obliviously simulating a RAM with N cells each of size $\mathbf{w} = \log N$ bits using an ORAM with cells of size \mathbf{b} bits and using local storage of size m bits such that $\mathbf{b}, m \in \text{poly } \log N$. There exists an offline ORAM scheme with $o(1)$ I/O efficiency, while every online ORAM scheme for this task must have $\Omega(\log N / \log \log N)$ I/O efficiency.*

We emphasize that the separation is *unconditional* in the sense that it neither assumes that schemes are in the balls and bins model (for the lower bound), nor that one-way functions exist (for the upper bound). Prior to this work, there was no such separation, even assuming either of these assumptions (and in any range of parameters).

1.2 Related Work

Passive Server. It is implicit in the standard definition of an ORAM that the server merely acts as a storage provider and does not perform any computation for the client. There are constructions where the server is actively performing computation (including memory I/O) for the client and this is not counted in the total I/O efficiency of the scheme (e.g., [1, 13, 19–21, 45, 52]). Many of these schemes achieve sub-logarithmic client-side I/O efficiency. Our lower bound shows that, in such cases, the server must have logarithmic I/O efficiency.

Related oblivious lower bounds. The beautiful result and technique of Larsen and Nielsen [33] inspired a fruitful line of works [26, 27, 32, 34, 42, 43]. Most related to the ORAM problem are [26, 27, 34, 43] on which we briefly elaborate. Jacob et al. [27] showed that the lower bound technique of [33] can be used to show logarithmic lower bounds on the overhead of oblivious simulation of various specific data structures like stacks, queues, and more. Persiano and Yeo [43] showed that logarithmic overhead is necessary for RAM simulation even if the security requirement is differential privacy, intuitively hiding only one

⁶ We believe that the $\log(\mathbf{b}/\mathbf{w})$ factor is necessary in the lower bound, at least for some range of parameters. Specifically, when $\mathbf{b}, m \in N^{\Theta(1)}$ and $\mathbf{w} = \log N$, by reparameterizing Path ORAM [50], we obtain an ORAM with $O(1)$ I/O efficiency.

access.⁷ Hubáček et al. [26] extended [33]’s logarithmic lower bound to the setting where the adversary does not see boundaries between queries. Larsen et al. [34] showed that logarithmic overhead in oblivious simulation is necessary even if data is allowed to be split over multiple servers, only one of which is controlled by an attacker.

All of the above papers give lower bounds that mostly apply to the symmetric setting where the cell size is identical in the given RAM and the simulated one since they suffer from a w/b factor loss. We believe that considering those problems and extending the lower bounds to the asymmetric setting (when possible) is intriguing, and we hope that our techniques in this paper will be helpful. In the full version [29], we show that using our techniques it is possible to improve the lower bound of Larsen et al. [34] to not suffer from a loss of w/b multiplicative factor even in the multi-server setting. This lower bound generalized our main result (Theorem 1) as it implies the latter when restricting to a single server. We refer to the full version [29] for the precise problem definition and statement of the result.

We believe that similarly, using our technique, one can improve the results in [26, 42] as they rely on a similar hard distribution to that of [33]. This is left for future work.

The cell probe model. Following Larsen and Nielsen [33], our lower bound holds in an augmented version of the well known cell probe model (to capture the obliviousness requirement). Details about our model are given in Sect. 3; Here, we mention some classical and notable facts about the cell probe model. The cell probe model, introduced by Yao [59], is a model of computation similar to the RAM model, except that all computational operations are free of charge except memory access. This model is useful in the analysis of data structures, especially for proving lower bounds on the number of memory accesses needed to solve a given problem.

By now, there are few techniques for proving lower bounds in the cell probe model. The strongest technique [31, 35] can prove super-logarithmic lower bounds and therefore should not be applicable as is to the ORAM setting where logarithmic upper bounds are known (unless additional requirements are made). Another technique, due to Pătraşcu and Demaine [44], is the so called *information transfer method* which is used to prove logarithmic lower bounds in the cell probe model. Larsen and Nielsen [33] were able to use this technique to prove their lower bound on ORAM constructions. We also use this technique. Persiano and Yeo’s [43] lower bound, mentioned above, were able to adapt the *chronogram* technique due to Fredman and Saks [18] which can also be used to prove logarithmic lower bounds.

Other related work. In the balls and bins model and where the server is passive (i.e., not performing any computation), Cash et al. [8] proved that any

⁷ The lower bound of Persiano and Yeo [43] also loses the w/b factor, similarly to Larsen and Nielsen. Specifically, it is $\Omega((w/b) \cdot \log(N/m))$ which is trivial if $b \gg w$. It is an open problem to improve their lower bound in the setting where $b \gg w$.

one-round ORAM must have either $\Omega(\sqrt{N})$ I/O efficiency or $\Omega(\sqrt{N})$ -bit local storage.

Boyle and Naor [7] proved that an unconditional lower bounds for offline ORAMs would imply a non-trivial circuit lower bound which is a long standing open problem. This result is obtained by constructing an offline ORAM from any sorting circuit, where the efficiency of the resulting ORAM is proportional to the size of the circuit. In a followup work, Weiss and Wichs [57] showed that proving a lower bound for *online read-only* ORAM is at least as hard as either proving a non-trivial circuit lower bound or ruling out a very good locally decodable code.

As mentioned, some ORAM constructions have improved I/O efficiency at the cost of setting the cell size \mathbf{b} to be super-logarithmic in the memory size. These works include not only schemes based on oblivious sorting [11, 23, 24, 52], but also several “tree-based” constructions [48, 50].

2 Technical Overview

This section gives a high level overview of our results. We first briefly recall the model and problem we want to solve. We proceed with explaining the beautiful technique of Larsen and Nielsen [33] and why it fails to give our desired lower bound. Lastly, building on the intuition we gained up to that point, we explain the main ideas in our proof and highlighting some of the technical challenges we are faced with.

2.1 The Model, Problem, and Recap of Larsen and Nielsen [33]

The model and problem. As observed by Larsen and Nielsen [33], it is convenient to state the ORAM problem as an oblivious data structure, as defined in [56], solving the *array maintenance* problem, where the goal is to maintain an array of N entries, each of size w bits, while supporting two operations: (1) (*write, a, x*): set the content of entry $a \in [N]$ to $x \in \{0, 1\}^w$ and (2) (*read, a*): return the content of entry $a \in [N]$. The lower bound that we prove, identical to [33], is on the *cell probe complexity* of any oblivious data structures solving the array maintenance problem. To get a lower bound on the I/O efficiency of ORAMs, it suffices to divide the number of probes by the number of operations.

Briefly, an oblivious data structure is a data structure that solves some given problem with an additional security guarantee which says that the (physical, observable) access patterns resulting from a sequence of logical data structure operations should reveal nothing on the latter sequence other than its length. For this purpose the oblivious data structure can use a small trusted/secure local storage (“cache”) on which it can perform operations “for free” and without leaking any data. The oblivious data structure is therefore parametrized by N', \mathbf{b}, m , its total number of cells, the bit-size of each cell, and the bit-size of its local storage, respectively. The efficiency metric of interest is the *number of probes* to the physical memory needed to answer one logical access. It is typically

assumed that $m \geq \mathbf{b} \geq \log N'$ so that the local storage can hold at least a single cell from the memory and that a single cell can hold a pointer to another cell.

Throughout most of this overview (except where we explicitly say otherwise), we consider the simpler setting where the oblivious data structure has *perfect* security and correctness. Perfect security means that for all sequence of logical operations of the same length, the observable sequence of physical memory probes is identically distributed. Perfect correctness means that the data structure never makes mistakes. With some additional technical work, these two assumptions can be relaxed.

Larsen and Nielsen’s lower bound. The lower bound of Larsen and Nielsen [33] adapts the *information transfer* technique of Pătraşcu and Demaine [44] to the oblivious setting. We give a high level overview next. Fix a given oblivious data structure for the array maintenance problem (i.e., an ORAM). For any sequence of N operations, we associate a complete binary tree with N leaves (we assume that N is a power of two for simplicity). The leaves are associated with the logical operations and their associated physical probes, in chronological order. That is, during the execution of the sequence, for each i , all cell addresses probed during the i th operation are associated with the i th leaf. Next, the leaf-level probes are *partially assigned* to internal nodes: for each probe to cell address q that is associated with leaf i , if chronologically the most recent probe to cell q happened during the j th operation (so that $j < i$), then the probe (i, q) is *assigned* to the lowest common ancestor of leaves i and j . Notice that the assignment is partial, i.e., some physical probes may not be assigned to any internal node, and thus it suffices to prove a lower bound on the total number of probes assigned to internal nodes.

For each fixed internal node v , Larsen and Nielsen [33] used the information transfer technique [44] to prove a lower bound on the number of associated physical probes with v by designing a hard distribution of sequences of operations. Let n be the number of leaves and thus operations in the subtree induced by v . In the hard distribution, all $N - n$ operations that are not in the subtree of v are just dummy reads from a fixed address. In the subtree induced by v , the first $n/2$ operations are writes to addresses $1, 2, \dots, n/2$ with uniformly random values $x_1, \dots, x_{n/2} \leftarrow \{0, 1\}^w$, and then the second $n/2$ operations are reads from addresses $1, 2, \dots, n/2$. That is,

$$(\text{write}, 1, x_1), \dots, (\text{write}, n, x_{n/2}), \quad (\text{read}, 1), \dots, (\text{read}, n/2).$$

To show that node v is associated with “many” probes when executing a sequence of operations from this distribution, the intuition is that in order to *correctly answer* the $n/2$ read operations, any data structure for the array maintenance problem (even non-oblivious ones!) must probe “many” cells that were also probed during the $n/2$ write operations. This intuition is formalized by a compression argument. Quantitatively, recalling that each cell in the array maintenance problem consists of w bits and each cell in the data structure consists of \mathbf{b} bits, there must exist a set of $\Omega(n \cdot w/\mathbf{b})$ cells from the data structure that are probed during the first as well as the second $n/2$ operations (here, we ignore

the local storage of m bits for simplicity). By the definition of our binary tree, all of these $\Omega(n \cdot \mathbf{w}/\mathbf{b})$ probes are associated with node v .

The proof proceeds by using the security guarantee of the data structure (as the above argument relied solely on correctness). The main observation is that since the tree and the associated probes of each node are efficiently computable by the adversary who only sees physical probes, then by security, the number of associated probes of each node must be the same for *all sequences of operations*. Namely, if node v is associated with $\Omega(n \cdot \mathbf{w}/\mathbf{b})$ probes when executing the hard distribution, then node v must also be associated with $\Omega(n \cdot \mathbf{w}/\mathbf{b})$ probes when executing any other sequence of operations of the same length; otherwise, an adversary can easily distinguish the two. Since the tree is a complete binary tree with N leaves, by summation there are $\Omega(N \cdot (\mathbf{w}/\mathbf{b}) \cdot \log N)$ associated probes to internal nodes which implies their lower bound.

Losing the \mathbf{w}/\mathbf{b} term is inherent when using the hard distribution designed by Larsen and Nielsen [33]. Recall that in their distribution we first write random values to addresses $1, \dots, n/2$ and then read those addresses in order. Indeed, using only correctness, each probe can carry information regarding \mathbf{b}/\mathbf{w} values and so the whole sequence of writes can be read using only $O(n \cdot \mathbf{w}/\mathbf{b})$ probes. The fundamental reason for the loss is therefore that the sequence of addresses in the read phase is completely determined a priori and the data structure can use this information during the write phase to organize data cleverly.

2.2 Our Hard Distribution and Information Transfer Tree

We propose the following hard distribution of sequences of $n + k \leq N$ operations. The first n operations are writes to addresses $1, 2, \dots, n$ with uniformly random values $x_1, \dots, x_n \leftarrow \{0, 1\}^w$ (same as in [33]). Then, in the last k operations, instead of sequentially reading from those addresses, we perform read from *uniformly random* words $a_1, a_2, \dots, a_k \leftarrow [n]$. That is,

$$(\text{write}, 1, x_1), \dots, (\text{write}, n, x_n), \quad (\text{read}, a_1), \dots, (\text{read}, a_k).$$

Indeed, now the sequence of reads is not known during the write phase so we avoid the aforementioned optimization the construction can use. But is this the only optimization? We prove that it is. The intuition is that no matter how large the cell size \mathbf{b} is, no matter how the data structure scheme processes the n write requests, in order to read from a uniformly random address $a_i \in [n]$ correctly, the construction must probe at least one cell (unless the construction got lucky and the corresponding value to address a_i was accidentally in the local storage). That is true only because the address a_i is chosen both randomly and online and therefore any pre-computation or pre-fetching that uses the fact that cells are moderately large is useless. In a high level, using a *compression argument* we show that for $k \leq n \cdot \mathbf{w}/\mathbf{b}$, the following holds:

Lemma: Any correct data structure solving the array maintenance problem when fed a length $n + k$ sequence of requests sampled from our hard

distribution, must probe $\Omega(k)$ cells during the **read** phase that were also probed during the **write** phase.

Whenever $\mathbf{b} \in \omega(\mathbf{w})$, this lower bound is better than the $\Omega(k \cdot \mathbf{w}/\mathbf{b})$ lower bound obtained with Larsen and Nielsen’s hard distribution. We note that we are only able to prove that the above statement holds with high-enough probability, smaller than 1 (which is enough to carry out the rest of the argument). Indeed, there will always be “easy” read sequences, like the one of Larsen and Nielsen, where the number of necessary probes will be smaller. Finally, we emphasize that in the above lemma, the **read** phase consists of only k operations (which differs from Larsen and Nielsen’s hard distribution which has n reads). This is specially designed to work with the information transfer tree that we will introduce below.

This lemma is central to our proof and while it may seem intuitively correct, the actual proof turns out to require very delicate and non-trivial probability analysis. We will get back to this in Sect. 2.3, where we will explain the main challenges and describe our solutions. Meanwhile, we proceed to explain how the lemma is used to derive the final lower bound using a generalized version of the information transfer tree described above.

Revisiting the information transfer tree. Recall that in the partial assignment of Larsen and Nielsen [33], a probe to a cell is assigned to a node v only if v is the lowest common ancestor between the probe and the most recent probe to the same cell. However, if a cell is probed 100 times during the **read** phase corresponding to v (i.e., v ’s right subtree), it will be counted and associated to v at most once! Working out the details, it turns out that even if we use our improved lemma from above in the binary tree approach, we would still lose the \mathbf{w}/\mathbf{b} factor. Therefore, we need to find a more fine-grained way to account for multiple probes to the same cell during the **read** phase.

Our solution is to consider a tree with larger arity so that we could count several probes to the same cell during the **read** phase of a given node (i.e., with multiplicity). We let χ , the arity of the tree, be proportional to \mathbf{b}/\mathbf{w} and consider a complete χ -ary tree with N leaves. Consider a node v that has an induced subtree of $2n$ leaves and consider an associated sequence of n writes followed by n reads. Divide the n read operations into $\chi/2$ equal-size groups so that each group has $k \triangleq n/(\chi/2)$ reads. For each such group we imagine a child node which is “in charge” of this group. Let the children of v that correspond to the **read** phase be $u_1, \dots, u_{\chi/2}$ so that each u_i is in charge of k disjoint **read** operations. Next in the partial assignment, we associate with v index-cell pairs of the form (i, q) , where i is an index from $[\chi/2]$ and q is a physical address of a probed cell. The index i tells us from which group the probe came and q tells us to which cell. Intuitively, this allows us to count probes to the same cell q with multiplicity, distinguishing them by the value of i . (In comparison, Larsen and Nielsen [33] only associated q ’s to nodes and so they do not distinguish multiple accesses to the same cell.) See Fig. 1 for an illustration.

Using our Lemma. Our lemma from above almost fits this framework. To prove that a group of k operations associated to node u_i introduces $\Omega(k)$ accesses that

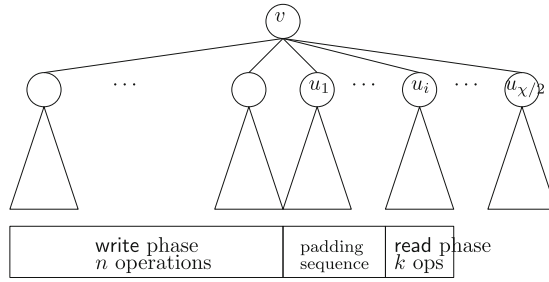


Fig. 1. Hard distribution on χ -ary tree.

are counted in v , we slightly modify the hard distribution to consist of a padding sequence of read operations (say from address 1) between the write phase and the reads that u_i is in charge of. Summing up over all u_i 's, the node v will be associated with $\Omega(\chi \cdot k) = \Omega(n)$ index-cell pairs, which is our goal and the best one can hope for.

The last step, where we use the obliviousness of the data structure in order to argue that any sequence of operations behaves as “the hardest one”, is similar to Larsen and Nielsen [33]. Recall that the tree is of depth $\log_\chi N$, the arity is χ , and for each level d , there are χ^d nodes at that level each has associated $\Omega(N/\chi^d)$ probes. Therefore, we get a lower bound of $\Omega(N \cdot \log N / \log(\mathbf{b}/\mathbf{w}))$ probes to perform N operations. This is essentially the lower bound claimed in Theorem 1, omitting the size of local storage m (which we ignored throughout this overview and only complicates the proof slightly).

Remark 1 (Relation to [44]). Pătraşcu and Demaine [44, Section 7] consider a related problem in a somewhat different context. There, they observe that the basic information transfer method suffers from the \mathbf{w}/\mathbf{b} factor loss. To remedy the situation they propose a new hard distribution, similar to ours, and also propose to consider an information transfer tree with higher arity, as we do. Essentially, our proof could be seen as an extension of their technique to the oblivious setting. The latter introduces many technical challenges, especially in the compression argument, as we elaborate next.

2.3 Our Compression Argument

Recall that our hard sequence consists of n writes to fixed addresses $1, \dots, n$ of uniformly random values followed by $k \leq n \cdot \mathbf{w}/\mathbf{b}$ reads from uniformly random addresses from $[n]$.⁸ Our goal is to argue that during the read phase, $\Omega(k)$ distinct cells must be probed. Let us refer to the write sequence as L and the read sequence

⁸ In fact, as mentioned we will need to consider an augmented sequence that has a padding sequence of reads from some fixed address in between the write sequence and the read sequence mentioned above. This will complicate the argument slightly so for simplicity we ignore it here.

as R (for left- and right-side). Denote by $\text{Cells}(L)$ the cells probed during the execution of the L sequence of accesses and by $\text{Cells}(R)$ the cells probed during the execution of the R sequence (after executing the L sequence). Note that $L, R, \text{Cells}(L), \text{Cells}(R)$ are all random variables. We want to prove that *with high probability* $|\text{Cells}(L) \cap \text{Cells}(R)| \in \Omega(k)$. That is, for some constant $\epsilon < 1$,

$$\Pr [|\text{Cells}(L) \cap \text{Cells}(R)| \geq \epsilon k] > 3/4, \quad (1)$$

where the probability is over the choice of L and R , and the randomness of the ORAM which influences $\text{Cells}(L)$ and $\text{Cells}(R)$.

The proof is done via a compression argument where we imagine two communication parties Alice and Bob. Alice gets as input $\mathbf{x} = x_1, \dots, x_n \leftarrow \{0, 1\}^w$ (chosen uniformly at random) and she sends one message to Bob who is able to recover \mathbf{x} . If the message sent by Alice contains $< n \cdot w$ bits, we get a contradiction. To this end, we assume that Inequality (1) is false, namely that the read phase can be implemented with ϵk probes for some small enough ϵ , and use that to get a too good to be true encoding scheme. This implies a contradiction, as needed. This proof is somewhat technical so we provide some intuition on how it works and refer to the technical section for full details.

Warmup: an expectation argument. It is insightful to first prove a weaker statement (which does not suffice for us) and then explain how to improve it. Here, we argue that

$$\mathbf{E} [|\text{Cells}(L) \cap \text{Cells}(R)|] \geq \epsilon k. \quad (2)$$

The proof is by contradiction, namely, we assume that Inequality (2) is false and obtain an impossible compression scheme. To this end, Alice and Bob share a long string \mathcal{S} that is chosen completely independent of the input to Alice. The string consists of (1) a sequence of k addresses $a_1, \dots, a_k \leftarrow [n]$ that define the R , (2) a random tape ρ for the ORAM, and (3) an integer $t \leftarrow [k]$ sampled uniformly at random. Note that even conditioned on the shared string \mathcal{S} , the entropy in the input to Alice, namely $x_1, \dots, x_n \leftarrow \{0, 1\}^w$, is still nw . Therefore, by Shannon's source coding theorem, the only way for Alice to correctly transmit them to Bob is by sending at least nw bits.

In a high level, Alice splits the indices $[n]$ into two groups: *easy* and *hard*. An index i is easy if Bob can learn value x_i *without* making a probe to $\text{Cells}(L)$, that is, a probe to a cell that was written to during the write sequence. All other indices are hard. By our assumption, the set of hard indices cannot be too large. Alice sends those hard values explicitly to Bob. To learn the values corresponding to easy indices, we use the correctness of the data structure to transfer them. The challenging part is for Alice to determine which index is easy and which is hard. Alice does this by seeing how likely it is to make the probe in $\text{Cells}(L)$ from a given index by "planting" that index in the random read operation given in \mathcal{S} (while keeping the rest of the operations fixed). If any $\text{Cells}(L)$ -probe occurs, this index is considered hard, otherwise it is easy. A more precise description follows.

Alice's encoding on input nw bits interpreted as $x_1, \dots, x_n \in \{0, 1\}^w$:

1. Using the ORAM, Alice executes the sequence of operations (L, R) prescribed by x_1, \dots, x_n and a_1, \dots, a_k . Then, Alice sends the contents of *overlapping cells* (yielded by the execution) to Bob, where the overlapping cells are defined as the cells probed during the write sequence L and then probed during the read sequence R (i.e., $\text{Cells}(L) \cap \text{Cells}(R)$).
2. For each $i \in [n]$, Alice replaces the t th read with operation (read, i) and (using the ORAM) executes the replaced sequence, that is, the sequence $(L, \widehat{R}_{t,i})$ where

$$L := \underbrace{(\text{write}, 1, x_1), \dots, (\text{write}, n, x_n)}_{\text{write phase}},$$

$$\widehat{R}_{t,i} := (\text{read}, a_1), \dots, (\text{read}, a_{t-1}), \underbrace{(\text{read}, i)}_{\text{planted read}}, (\text{read}, a_{t+1}), \dots, (\text{read}, a_k).$$

Depending on the probed locations induced by (read, i) , do:

- (a) If (read, i) probes at least one cell that was written to during the write phase (i.e., in $\text{Cells}(L)$), then i is called *hard*. Alice sends value (i, x_i) directly to Bob.
- (b) Otherwise, (read, i) probes no cell in $\text{Cells}(L)$ and i is called *easy*. Alice sends nothing to Bob as Bob can recover x_i by executing (read, i) himself.

On Bob’s side, the hard x_i ’s are received from Alice directly, while the easy x_i ’s are recovered by executing (read, i) planted as the t th read operation, that is, after the prefix $(\text{read}, a_1), \dots, (\text{read}, a_{t-1})$. Bob indeed recovers all easy x_i ’s correctly: Bob received the content of the overlapping cells that suffice to execute the prefix read sequence.

Analyzing the size of the message from Alice to Bob is a bit more challenging. In a high level, Alice’s message consists of just two parts, the contents of overlapping cells and the values of “hard” inputs. By assumption (Inequality (2) is false), the number of overlapping cells is ϵk and so the first part consists of at most $\epsilon k b \leq \epsilon n w$ bits. For the second part, roughly speaking, we consider all possible samples of $(a_1, a_2, \dots, a_k, t) \in [n]^k \times [k]$ while fixing x_1, \dots, x_n . By assumption, with probability at most ϵ , (read, a_t) is hard, which means that at most ϵ fraction of all such samples are hard. Then, for any set of n distinct samples, on average, there are at most ϵn hard samples. Noticing that Alice’s procedure is choosing a random set of n samples, we conclude that *in expectation* there are ϵn hard samples, which means ϵn hard x_i ’s on average. It follows that the second part of Alice’s message consumes $\epsilon n w$ bits, and then the total message length is $2\epsilon n w$ bits, which is a contradiction when ϵ is small enough.

The high probability argument. Recall that in the last step of lower bound proof we need to move from a claim about the load of a node in the information transfer tree to the load of the same node under any other input sequence of operations. Since security only holds with constant probability, this step loses a constant factor and therefore we need our original compression argument to hold *with high probability* and not just in expectation.

This complicates the compression argument as follows. Now, Alice cannot just send the content of the overlapping cells directly to help Bob answer easy queries (for which it uses the correctness of the data structure), since there is no bound on the expected number of overlapping cells. Instead, we modify Alice’s procedure to distinguish between two cases, either sending the overlapping cells directly is too expensive or it is not. In the latter case, we need to analyze and bound the number of hard indices i *conditioned* on the event that the number of overlapping cells is small. This requires delicate conditional probability analysis on which we elaborate next. In the former case, there is no compression since Alice just sends all x_1, \dots, x_n in the clear but we can show that this case does not happen too often due to the assumption (Inequality (1) is false).

Specifically, the most challenging is to prove that *conditioned on the overlapping cells set being small*, the expected size of the set of hard indices is bounded by a sufficiently small constant times n . Let $\text{Good}_{L,R}$ be the conditioned event. What we show is that if $\beta < 3/4$ is a constant for which $\Pr[|\text{Cells}(L) \cap \text{Cells}(R)| \geq \epsilon k] = \beta$ (our assumption, see Inequality (1)), then:

$$\text{Lemma: } \mathbf{E}[|H| \mid \text{Good}_{L,R}] < (\beta + \epsilon/(1 - \beta))n.$$

We define $\text{Good}_{L, \widehat{R}_{t,i}}$ similarly as the event when the overlapping cells between $(L, \widehat{R}_{t,i})$ is small. By linearity of expectation and the law of total probability:

$$\begin{aligned} \mathbf{E}[|H| \mid \text{Good}_{L,R}] &= \sum_{i \in [n]} \Pr[i \in H \mid \text{Good}_{L,R}] \\ &= \sum_{i \in [n]} \Pr[i \in H \wedge \neg \text{Good}_{L, \widehat{R}_{t,i}} \mid \text{Good}_{L,R}] \\ &\quad + \sum_{i \in [n]} \Pr[i \in H \wedge \text{Good}_{L, \widehat{R}_{t,i}} \mid \text{Good}_{L,R}]. \end{aligned}$$

We now bound each of these terms separately. It is rather easy (though a bit technical) to bound the second term. Specifically, we show that $\sum_{i \in [n]} \Pr[i \in H \wedge \text{Good}_{L, \widehat{R}_{t,i}} \mid \text{Good}_{L,R}] \leq \epsilon n/(1 - \beta)$. Indeed, for each $i \in [n]$, $\Pr[i \in H \wedge \text{Good}_{L, \widehat{R}_{t,i}} \mid \text{Good}_{L,R}] \leq \Pr[i \in H \wedge \text{Good}_{L, \widehat{R}_{t,i}}] / \Pr[\text{Good}_{L,R}]$. So, the denominator is exactly $1 - \beta$. The fact that the nominator is bounded by ϵ follows from the definition of $\text{Good}_{L, \widehat{R}_{t,i}}$.

The bound on the first term is much more interesting. In words, the event we are trying to bound corresponds to sampling the sequences L and R and then $\widehat{R}_{t,i}$ and asking what is the probability that $\text{Good}_{L, \widehat{R}_{t,i}}$ occurs *conditioned* on $\text{Good}_{L,R}$ occurring (ignoring event $i \in H$). To analyze this event, we recall that $\widehat{R}_{t,i}$ is obtained by *resampling* the t th operation in R . So, what is the probability that by resampling only one read operation in R we suddenly do not satisfy the event Good ? We prove a general lemma that *partial resampling* cannot reduce the probability beyond a certain point! Here is a simple variant of the lemma (we state and prove a more general version in the full version [29]):

Partial Resampling Lemma: Consider two independent random variables X and Y . Let Y^* be an independent random variable distributed identically to Y . Let f be an arbitrary Boolean function. Then,

$$\Pr[f(X, Y^*) = 1 \mid f(X, Y) = 1] \geq \Pr[f(X, Y) = 1].$$

This means that if the event $\text{Good}_{L,R}$ occurs, then it must also occur in $\text{Good}_{L, \widehat{R}_{t,i}}$ with good probability. Plugging in the assumption, we can bound the second term by βn .

Together, the two bounds imply that $\mathbf{E}[|H| \mid \text{Good}_{L,R}] < (\beta + \epsilon/(1 - \beta))n$, as needed.

3 The Model

This section introduces the model in which our lower bound is proven. As in previous works [27, 33, 43], we start-off with the cell probe model, first described by Yao [59]. Traditionally, this model is used to prove lower bounds for word-RAM data structures and is extremely powerful in the sense that it allows arbitrary computations and only charges for memory accesses.

In a high-level, the cell probe model models the interaction between a CPU and a memory. The memory is modeled as a word-RAM, that is, an array of cells such that each cell can contain at most b bits. The CPU can perform operations on the memory, namely, either reading the content of some cell or overwriting the content of some cell. An algorithm executed in this setting is charged one unit of cost on every operation it makes (read or write) and all computation based on the contents of probed cells is free of charge.

Whereas this model captures traditional data structures, it does not capture data structures that have privacy requirements for the stored data and/or the operations performed. Indeed, the latter are usually modeled in the client-server model, where a client wishes to outsource data to server while retaining the ability to perform computation over the data. At the same time, the client wishes to hide the performed operations as well as the contents of its data cells from the server who sees the entire memory and the memory accesses. To address this gap, Larsen and Nielsen [33] introduced the *Oblivious Cell Probe Model*, an augmented version of the cell probe model. We briefly introduce this model next, mostly following Larsen and Nielsen.

Data structure problems. A data structure problem in the oblivious cell probe model is defined by a tuple $(\mathcal{U}, \mathcal{Q}, \mathcal{O}, f)$, where \mathcal{U} is a universe of update operations, \mathcal{Q} is a universe of queries, and \mathcal{O} is an output domain. Furthermore, there is a query function $f: \mathcal{U}^* \times \mathcal{Q} \rightarrow \mathcal{O}$. For a sequence of updates $u_1, \dots, u_M \in \mathcal{U}$ and a query $q \in \mathcal{Q}$, we say that the answer to the query q after updates u_1, \dots, u_M is $f(u_1, \dots, u_M, q)$.

Oblivious Cell Probe Data Structures. An oblivious cell probe data structure for a given data structure problem $\mathcal{P} = (\mathcal{U}, \mathcal{Q}, \mathcal{O}, f)$, consists of a randomized algorithm implementing the update and query operations for \mathcal{P} . The data

structure is parametrized by three integers m , \mathbf{b} , and N' , denoting the client storage and cell size in bits, and the number of cells respectively. We follow the standard assumption $\log N' \leq \mathbf{b}$ so that any cell can store the address of any other cell. We further assume that the data structure has access to a finite string of randomness ρ of length ℓ . The parameter ℓ can be arbitrary large and so ρ can contain a random oracle. Fixing ρ , the algorithm DS is deterministic. As such, the data structure can be described by a decision tree T_{op} for every operation $\text{op} \in \mathcal{U} \cup \mathcal{Q}$, i.e., it has one decision tree for every possible operation in the data structure problem. Each node in the decision tree is labelled by an index indicating the location to probe in the memory (held by the server). The decision of which path to continue to in the tree depends on the answer to the probe to the memory and small local information stored by the client.

More precisely, each node in the decision tree T_{op} , where $\text{op} \in \mathcal{U} \cup \mathcal{Q}$, is labeled by an address $i \in [N']$ and it has one child for every triple of the form $(m_0, c_0, \rho) \in \{0, 1\}^m \times \{0, 1\}^{\mathbf{b}} \times \{0, 1\}^{\ell}$. Each edge to a child is further labeled by $(j, m_1, c_1) \in [N'] \times \{0, 1\}^m \times \{0, 1\}^{\mathbf{b}}$. To process an operation op , the oblivious cell probe data structure starts its execution at the root of the tree and traverses from root to leaf. When visiting a node v in this traversal, labelled with some address $i_v \in [N']$, it probes the memory cell i_v . If C denotes its content, M denotes the current contents of the client memory and ρ denotes the random bit-string, the process continues by descending to the child of v corresponding to the tuple (M, C, ρ) . If the edge to the child is labelled (j, m_1, c_1) , then the memory cell of address j has its contents updated to c_1 and the client memory is updated to m_1 . We say that memory cell j is *probed*. The execution stops when reaching a leaf. Each leaf v of the decision tree T_{op} , where $\text{op} \in \mathcal{Q}$, is labeled with an element ans_v in \mathcal{O} (the answer to the query). We say that the oblivious cell probe data structure returns ans_v as its answer to the query op .

I/O efficiency. The I/O efficiency of an oblivious data structure is related to the depth of the decision tree as each edge corresponds to a cell probe. Furthermore, our model assumes that the server is passive, i.e., it can only update or retrieve a cell for the client.

Definition 1 (Expected amortized I/O efficiency). *An oblivious cell probe data structure has expected amortized I/O efficiency $t(M)$ on a sequence y of M operations from $\mathcal{U} \cup \mathcal{Q}$ if the total number of memory probes is no more than $t(M) \cdot M$ in expectation. The expectation is taken over the random choice of the randomness $\rho \in \{0, 1\}^{\ell}$. An oblivious cell probe data structure has expected amortized I/O efficiency $t(M)$ if it has expected amortized I/O efficiency $t(M)$ on all sequences y of operations from $\mathcal{U} \cup \mathcal{Q}$.*

Remark 2 (Other efficiency notions). There are few other metrics of efficiency of interest in the context of ORAM constructions. It is common to consider the *bandwidth* efficiency of a construction, namely, the communication complexity consumed by the construction when processing a sequence of operations, amortized per operation. This is equal to \mathbf{b} times the I/O efficiency. Vice versa, if the

amortized bandwidth of a construction is $t(\cdot)$, then the I/O efficiency of that construction is t/\mathbf{b} .

Thus, there is a $Q = Q(N, \mathbf{b}, \mathbf{w})$ lower bound on I/O efficiency if and only if there is a $\mathbf{b} \cdot Q$ lower bound on bandwidth. For example, suppose that $\mathbf{w} = \log N$ and $\mathbf{b}, m \in \Theta(\log^2 N)$. Then, the previously known lower bound [33] says that $\Omega(\log^2 N)$ amortized bandwidth is necessary (that is $\Omega(1)$ I/O efficiency), but our improved lower bound says that $\Omega(\log^3 N / \log \log N)$ bandwidth is necessary (that is $\Omega(\log N / \log \log N)$ I/O efficiency).

It is also common to measure the complexity of an ORAM construction in the language of efficiency *overhead* (either I/O or bandwidth) where we compare the ratio between the efficiency of the ORAM and the efficiency of the insecure RAM. This makes complete sense when $\mathbf{b} = \mathbf{w}$, but when $\mathbf{b} \in \omega(\mathbf{w})$ it is more confusing since the basic unit of cost (cell size) is different between the two settings. Some papers do explicitly distinguish between \mathbf{w} and \mathbf{b} [48, 49, 52, 57] and measure complexity correctly. For clarity, we will avoid the term *overhead*.

Correctness and security. Let $y = (\text{op}_1, \dots, \text{op}_M)$ be a sequence of M operations to the given data structure problem, where each $\text{op}_i \in \mathcal{U} \cap \mathcal{Q}$. For an oblivious cell probe data structure, define the (possibly randomized) probe sequence on y as the tuple:

$$\text{Access}(y) = (\text{Access}(\text{op}_1), \dots, \text{Access}(\text{op}_M)),$$

where $\text{Access}(\text{op}_i)$ is the sequence of memory addresses probed while processing op_i . More precisely, let $\text{Access}(y; \rho) := (\text{Access}(\text{op}_1; \rho), \dots, \text{Access}(\text{op}_M; \rho))$ be the deterministic sequence of operations when the random bit-string fixed to ρ and let $\text{Access}(y)$ be the random variable describing $\text{Access}(y; \rho)$ for a random $\rho \in \{0, 1\}^\ell$.

Definition 2 (Correctness and security). *An oblivious cell probe data structure is said to be δ -correct and ϵ -secure if the following two properties hold:*

- **Security:** *For any two data request sequences y and z of the same length M , their probe sequences $\text{Access}(y)$ and $\text{Access}(z)$ cannot be distinguished with probability better than ϵ by an algorithm which is polynomial time in $M + \log |\mathcal{U}| + \log |\mathcal{Q}| + \mathbf{b}$.*
- **Correctness:** *The oblivious cell probe data structure has failure probability at most δ , namely, for every sequence and any operation op in the sequence, the data structure answers op correctly with probability at least $1 - \delta$.*

ORAM is array maintenance. As observed in previous work [33], the definition of an online ORAM coincides with the definition of an oblivious data structure (see [56]) solving the *array maintenance problem*. In this problem, the goal is to maintain an array of N entries, each of size \mathbf{w} bits, while allowing write and read operations, where (write, i, a) sets the content of the i th cell to the value a and (read, i) return the content of the i th cell (for $i \in [N]$ and $a \in \{0, 1\}^\mathbf{w}$).

Therefore, in order to prove a lower bound on the I/O efficiency of an ORAM scheme, it suffices to prove a lower bound on the I/O efficiency of any correct and secure data structure for the array maintenance problem in the oblivious cell probe model.

Remark 3 (Operation boundaries). We follow Larsen and Nielsen [33] and assume that the adversary sees which cell access belongs to which operation from y . Hubáček et al. [26] were able to extend the lower bound of Larsen and Nielsen [33] to account for this gap. We suspect that our techniques and lower bound could be extended to capture this stronger setting, but it is left for future work.

4 An ORAM Lower Bound

This section is devoted to the proof of our lower bound on the I/O efficiency of oblivious cell probe data structures solving the array maintenance problem. As mentioned, such a lower bound directly implies an I/O efficiency lower bound for online ORAMs. Our main theorem is stated next.

Theorem 3 (Main theorem). *Let \mathcal{DS} be an oblivious cell probe data structure for the array maintenance problem on arrays of N entries, each of size \mathbf{w} bits. Let N' denote the number of cells in \mathcal{DS} , \mathbf{b} denote the cell size in bits, and m denote the number of bits of client memory. Assume that $16 \leq \mathbf{w} \leq \mathbf{b}$ and $\mathbf{w} \leq m \leq N\mathbf{w}$.*

If \mathcal{DS} is $(1/128)$ -correct and $(1/4)$ -secure, then there is a sequence of $\ell \in (N/(2 \lceil \mathbf{b}/\mathbf{w} \rceil), N]$ operations such that the expected amortized I/O efficiency of \mathcal{DS} on this sequence is

$$\Omega \left(\frac{\log(N\mathbf{w}/m)}{1 + \log \lceil \mathbf{b}/\mathbf{w} \rceil} \right).$$

In particular, when $\mathbf{w} \leq m \leq N^{1-\epsilon}$ for $\epsilon > 0$, $\mathbf{b} = \log^c N$ for $c > 1$, and $\mathbf{w} = \log N$, the I/O efficiency is $\Omega \left(\frac{\log N}{\log \log N} \right)$. The rest of this section is devoted to the proof of Theorem 3.

Proof (Proof of Theorem 3). We start with the following definition.

Definition 3 (Set of probed cells). *Given a length M sequence of operations, $\text{seq} = (\text{op}_1, \dots, \text{op}_M)$, define $\text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})$ as the set of addresses of (physical) cells accessed by \mathcal{DS} during its execution of operation op_i after executing the sequence $(\text{op}_1, \dots, \text{op}_{i-1})$. Similarly, given seq and $i, j \in [M]$ such that $i < j$, $\text{Cells}(\text{op}_i, \text{op}_{i+1}, \dots, \text{op}_j \mid \text{op}_1, \dots, \text{op}_{i-1})$ is defined as the set of addresses of cells accessed by \mathcal{DS} during its execution of operations $(\text{op}_i, \dots, \text{op}_j)$ after executing the sequence $(\text{op}_1, \dots, \text{op}_{i-1})$.*

Notice that we define $\text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})$ as a set and so its cardinality does not account for multiplicities. Therefore, we will use the sum of cardinalities $\sum_{i \in [M]} |\text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})|$ as a lower bound on the total number of accesses made by \mathcal{DS} .

We now construct the information transfer tree. Fix ℓ to be a power of $\chi := 2 \lceil \mathbf{b}/\mathbf{w} \rceil$ in the range $(N/(2 \lceil \mathbf{b}/\mathbf{w} \rceil), N]$. Let T be the complete χ -ary tree consisting of ℓ leaves (see Fig. 2 for visualization). For any sequence of operations $\text{seq} = (\text{op}_1, \dots, \text{op}_\ell)$, for each $i \in [\ell]$, we associate op_i to the i th leaf of T . Additionally, $\text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})$ (i.e., the addresses of cells accessed by \mathcal{DS} during its execution of the i th operation in the sequence) are associated to the same i th leaf. For each accessed cell q that is associated with a leaf i , we map q to at most one internal node v of T , where v is an ancestor of i . This is described next.

First, for each internal $v \in \mathsf{T}$, we define a set of index-cell pairs, $P_v(\text{seq})$, as follows. A pair of index-cell $(i, q) \in [\ell] \times [N']$ is in $P_v(\text{seq})$ if and only if

- i is a leaf in the subtree induced by v and $q \in \text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})$,
- There exists $j < i$ such that $q \in \text{Cells}(\text{op}_j \mid \text{op}_1, \dots, \text{op}_{j-1})$,
- For all $j' \in \{j + 1, \dots, i - 1\}$, it holds that $q \notin \text{Cells}(\text{op}_{j'} \mid \text{op}_1, \dots, \text{op}_{j'-1})$, and
- The lowest common ancestor of i and j is v .

Notice that each cell access $q \in \text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})$ during the execution of op_i is assigned to at most one $v \in \mathsf{T}$. Hence, for any seq and execution of \mathcal{DS} , we have that

$$\sum_{i \in [\ell]} |\text{Cells}(\text{op}_i \mid \text{op}_1, \dots, \text{op}_{i-1})| \geq \sum_{v \in \mathsf{T}} |P_v(\text{seq})|.$$

We conclude the proof of the theorem using the following lemma whose proof is given below.

Lemma 1. *Let $\epsilon := 1/128$. Fix any sequence seq consisting of ℓ operations. Let $v \in \mathsf{T}$ be an internal node whose subtree consists of at least $2 \cdot \max\{8, m/(\epsilon \mathbf{w})\}$ leaves. For any $(1/4)$ -secure and $(1/128)$ -correct \mathcal{DS} against ℓ operations, it holds that*

$$\mathbf{E}[|P_v(\text{seq})|] \geq \epsilon \cdot \ell / (4\chi^{d(v)}),$$

where $d(v)$ is the depth of v (i.e. the distance from v to the root).

Let us first explain why Lemma 1 implies Theorem 3. Let d^* be the maximum depth for which Lemma 1 applies. Summing over all nodes in T , by linearity of expectation, we have that

$$\mathbf{E} \left[\sum_{v \in \mathsf{T}} |P_v(\text{seq})| \right] = \sum_{v \in \mathsf{T}} \mathbf{E}[|P_v(\text{seq})|] \geq \sum_{v \in \mathsf{T}, d(v) \in [0, d^*]} \mathbf{E}[|P_v(\text{seq})|] \geq (d^* + 1) \cdot \epsilon \ell / 4,$$

where the last inequality follows by Lemma 1. Since Lemma 1 applies to any node v that has at least $2 \cdot \max\{8, m/(\epsilon \mathbf{w})\}$ leaves in its induced subtree, we have

$$d^* := \left\lceil \log_\chi \left[\frac{\ell}{2 \cdot \max\{8, m/(\epsilon \mathbf{w})\}} \right] \right\rceil \in \Omega \left(\frac{\log(N\mathbf{w}/m)}{1 + \log(\mathbf{b}/\mathbf{w})} \right)$$

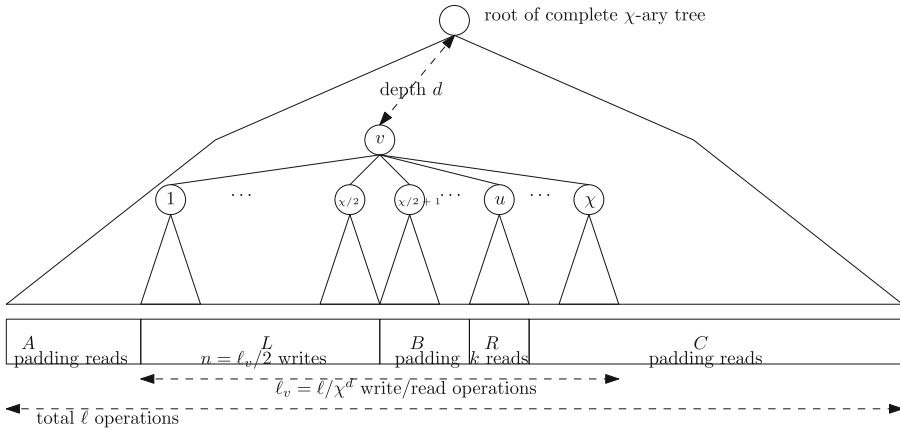


Fig. 2. The distribution $\mathcal{D}(v, u)$ of hard sequences for the parent-child pair (v, u) in the complete χ -ary tree of ℓ leaves. Each leaf is associated with a read or write operation, and the hard sequence is the operations from the left-most to the right-most leaves. Given the internal node v and its child u where u is in the right-side of the subtree induced by v , we focus on the operations in the Left-side of the subtree of v , i.e., L part, and on the operations in the subtree induced by u , i.e., R (for right-side) part. The L part is $n = \ell_v/2$ write operations to fixed locations with random contents (where $\ell_v = \ell/\chi^d$ is the number of leaves in the subtree of v), and the R part is $k = \ell_v/\chi$ read operations from random locations that were written in L part. The remaining parts A, B, C are all padding operations that just read the fixed location 1. The overall hard sequence is then (A, L, B, R, C) .

for all $m, \mathbf{b}, \mathbf{w}, N$ such that $\mathbf{b} \geq \mathbf{w} \geq 16$ and $\mathbf{w} \leq m \leq N\mathbf{w}$ (which ensure that the logs are nonnegative). Hence, for any seq of ℓ operations, the expected number of accesses is lower bounded by $\ell \cdot \Omega\left(\frac{\log(N\mathbf{w}/m)}{1+\log(\mathbf{b}/\mathbf{w})}\right)$, which concludes the proof of Theorem 3.

We conclude this section with the proof of Lemma 1. Note that this proof will rely on Theorem 5 which is stated and proved in Sect. 5.

Proof (Proof of Lemma 1). Recall that $P_v(\text{seq})$ consists of pairs of index-cell pairs (i, q) such that during the i th operation \mathcal{DS} accesses physical cell q and also the most recent access to q was made at some operation $j < i$ such that j is a leaf in the induced subtree of v and v is the the lowest common ancestor of i and j . Denote $P_{v,u}(\text{seq})$ the subset of (i, q) in $P_v(\text{seq})$ that result from an operation i that happens in the subtree induced by u . It holds that

$$|P_v(\text{seq})| = \sum_{u \text{ is a child of } v} |P_{v,u}(\text{seq})|. \tag{3}$$

We therefore prove a lower bound on each $|P_{v,u}(\text{seq})|$. To this end, for a given pair of parent-child, (v, u) , in the tree, we design a distribution of access seq_{hard} which causes $|P_{v,u}(\text{seq}_{\text{hard}})|$ to be large with high probability. We then

use the security guarantee of \mathcal{DS} , ensuring that the access pattern resulting from executing any `seq` must be indistinguishable, and therefore the same large number of probes must occur on any input sequence. That is, $|P_{v,u}(\text{seq})|$ is large with high probability. We give the hard distribution next.

The hard distribution. To describe the distribution of hard sequences, we set up some notation. Specifically, we will explain how to “split” a given length ℓ sequence of operations w.r.t a given internal node $v \in \mathbb{T}$.

- Let $d := d(v)$ be the depth of the node v , and let $l := l(v) \in [\chi^d]$ be the index of v in the d th level.
- Let $\ell_v := \ell/\chi^d$ be the number of leaves in the subtree induced by v . Set $n := \ell_v/2$, and $k := \ell_v/\chi$.
- Recall that v has χ children. Let $U := \{\chi/2 + 1, \chi/2 + 2, \dots, \chi\}$ be the set of indices of second half children of v (i.e., the right half of children). Given $u \in U$, we slightly abuse notation and say that the u th child of v is u .

Because our goal is to bound the number of probes during the subtree of u , we choose to perform n writes during the first n leaves of v , and then perform k reads during the k leaves of $u \in U$ (Fig. 2). The remaining parts are just padding to ℓ operations. Formally, the distribution of hard sequence $\mathcal{D}(v, u)$, with induced parameters l, ℓ_v, n, k as above, is sampled as follow:

1. Let A be the sequence consisting of $(l - 1) \cdot \ell_v$ dummy reads, i.e., repeating $(\text{read}, 1)$ for $(l - 1) \cdot \ell_v$ times.

$$A := \underbrace{(\text{read}, 1), \dots, (\text{read}, 1)}_{(l-1) \cdot \ell_v \text{ times}}$$

2. Let L (for left-side) be the sequence of n writes to fixed locations with random words, i.e.,

$$L := (\text{write}, 1, x_1), (\text{write}, 2, x_2), \dots, (\text{write}, n, x_n),$$

where $x_1, \dots, x_n \leftarrow \{0, 1\}^w$ are chosen independently uniformly at random.

3. Let B be the sequence consisting of $k \cdot (u - 1) - n$ dummy reads,

$$B := \underbrace{(\text{read}, 1), \dots, (\text{read}, 1)}_{k \cdot (u-1) - n \text{ times}}$$

4. Let R (for right-side) be the sequence of k reads from random addresses in $[n]$, i.e.,

$$R := (\text{read}, a_1), (\text{read}, a_2), \dots, (\text{read}, a_k),$$

where $a_1, \dots, a_k \leftarrow [n]$ are chosen independently uniformly at random.

5. Let C be the sequence of dummy reads whose goal is to pad the whole sequence to length ℓ ,

$$C := \underbrace{(\text{read}, 1), \dots, (\text{read}, 1)}_{\ell - (l-1) \cdot \ell_v - u \cdot k \text{ times}}$$

* **Output** the concatenated length ℓ sequence

$$\text{seq}_{\text{hard}} = A, L, B, R, C.$$

We are interested in the set of cells that are touched both during the L, B sequence and during the R sequence, i.e., the set $\text{Cells}(L, B \mid A) \cap \text{Cells}(R \mid A, L, B)$ (see Definition 3 for $\text{Cells}(\dots)$ notation). By definition, it holds that

$$|P_{v,u}(\text{seq}_{\text{hard}})| \geq |\text{Cells}(L, B \mid A) \cap \text{Cells}(R \mid A, L, B)|.$$

In Theorem 5 we prove the following.

Theorem 4 (See Theorem 5). *Let $\delta := 1/128$ and $\epsilon := 1/128$. If \mathcal{DS} is δ -correct (for the array maintenance problem), then as long as $n \in [\max\{8, m/(\epsilon w)\}, N]$ and $k \leq n \cdot w/b$, it holds that*

$$\Pr [|\text{Cells}(L, B \mid A) \cap \text{Cells}(R \mid A, L, B)| \geq \epsilon k] > 3/4.$$

Indeed, observe that the conditions to apply this theorem are met since $k \leq n w/b$ as $n = \ell_v/2$, $k = \ell_v/\chi$, and $\chi = 2 \lceil b/w \rceil$. Also, since v is an internal node whose induced subtree consists of $\ell_v \geq 2 \cdot \max\{8, m/(\epsilon w)\}$ leaves, we also have $n \in [\max\{8, m/(\epsilon w)\}, N]$. Therefore, $\Pr [|P_{v,u}(\text{seq}_{\text{hard}})| \geq \epsilon k] > 3/4$.

Due to the security guarantee of \mathcal{DS} , we deduce that for any (equal-length) sequence seq the above should hold. Namely, denoting the randomness of the \mathcal{DS} by ρ , we have

$$\Pr_{\text{seq}_{\text{hard}}, \rho} [|P_{v,u}(\text{seq}_{\text{hard}})| \geq \epsilon k] - \Pr_{\rho} [|P_{v,u}(\text{seq})| \geq \epsilon k] \leq 1/4.$$

Therefore, we obtain that $\Pr [|P_{v,u}(\text{seq})| \geq \epsilon k] > 1/2$ and so $\mathbf{E} [|P_{v,u}(\text{seq})|] > \epsilon k/2$. Using Eq. (3) and linearity of expectation we obtain that

$$\begin{aligned} \mathbf{E} [|P_v(\text{seq})|] &= \mathbf{E} \left[\sum_{u \text{ is a child of } v} |P_{v,u}(\text{seq})| \right] \\ &= \sum_{u \text{ is a child of } v} \mathbf{E} [|P_{v,u}(\text{seq})|] \\ &> (\chi/2) \cdot (\epsilon k/2) = \epsilon \ell / (4\chi^d). \end{aligned}$$

5 The Compression Argument

Let \mathcal{DS} be an oblivious cell probe data structure for the array maintenance problem on arrays of N entries, each of w bits. Let N' denote the number of cells in \mathcal{DS} , let b denote the bit-length of each cell, and let m denote the number of bits of client memory.

Consider the following distribution over sequences of operations given to \mathcal{DS} . The distribution is denoted $\mathcal{D}_{A,B,n,k}$ and it is parametrized by two sequences of operations A and B , and by two positive integers $n, k \leq N$. The sequence

A consist of arbitrary reads and writes (A is going to be a prefix sequence) and B consist of arbitrary reads but *no* writes (B is going to be a padding sequence). Each sequence of operations sampled from $\mathcal{D}_{A,B,n,k}$ consists of 4 parts, A, L, B, R , in this order, where L (for left-side) is a sequence of n writes to fixed addresses $1, \dots, n$ with uniformly random data, and R (for right-side) is a sequence of k reads from uniformly random indices in $[n]$. The full sequence (A, L, B, R) looks as follows:

- A : Fixed sequence of reads and writes;
- L : (write, 1, x_1), (write, 2, x_2), \dots , (write, n , x_n),
 where $x_1, \dots, x_n \leftarrow \{0, 1\}^w$, chosen uniformly at random;
- B : Fixed sequence of reads;
- R : (read, a_1), (read, a_2), \dots , (read, a_k),
 where $a_1, \dots, a_k \leftarrow [n]$, chosen uniformly at random.

Recall that in Definition 3, given a sequence of operations (X, Y) and randomness ρ , we let $\text{Cells}(Y \mid X)$ be the set of addresses of (physical) cells probed by \mathcal{DS} during its execution of the Y sequence *after* executing the X sequence.⁹ For example, in an instance of sequence (A, L, B, R) sampled from our distribution, (1) $\text{Cells}(L, B \mid A)$ contains the (physical) addresses of cells probed by \mathcal{DS} during the execution of the L and B parts after executing the A sequence, and (2) $\text{Cells}(R \mid A, L, B)$ contains the (physical) addresses of cells probed by \mathcal{DS} during the execution of the R sequence after executing the A, L , and B sequences. We prove the following theorem.

Theorem 5. *Let $\delta := 1/128$, $\epsilon := 1/128$ and $\alpha := 3/4$. Further, fix integers $n \in [\max\{8, m/(\epsilon w)\}, N]$, $w \geq 16$, and $k \leq n \cdot w/b$. Lastly, fix arbitrary sequences A and B as above. Then, if \mathcal{DS} is δ -correct (for the array maintenance problem), then it holds that*

$$\Pr [|\text{Cells}(L, B \mid A) \cap \text{Cells}(R \mid A, L, B)| \geq \epsilon k] > \alpha,$$

where the probability is taken over the choice of L and R (i.e., over the choice of (A, L, B, R) from $\mathcal{D}_{A,B,n,k}$), and over the internal randomness of \mathcal{DS} .

In order to prove Theorem 5, we assume for contradiction that the statement is false, namely that there are A, B, n, k as in the theorem statement and a $\beta \leq \alpha$ for which

$$\Pr [|\text{Cells}(L, B \mid A) \cap \text{Cells}(R \mid A, L, B)| \geq \epsilon k] = \beta. \tag{4}$$

To reach a contradiction, we construct a randomized compression scheme that encodes nw uniformly random bits into a message that is less than nw bits. Section 5.1 describes the encoding and decoding procedure of such compression,

⁹ Notice that $\text{Cells}(Y \mid X)$ is a *set* of addresses, whereas $\text{Access}(X \parallel Y)$ is a *sequence* of addresses.

and it also shows the compression is correct. We then in Sect. 5.2 prove that the expected size of the encoding is less than nw bits, which is a contradiction to Shannon’s source coding theorem and concludes the proof of Theorem 5.

The reader may find it helpful to first read the full version [29] where we prove a weaker version of Theorem 5. Specifically, we show that the *expected* size of the intersection of both sets from Theorem 5 is $\Omega(k)$ (rather than that it holds with high probability).

5.1 The Encoding and Decoding Procedures

The encoder, Alice, gets as input the nw random bits interpreted as $x_1, \dots, x_n \in \{0, 1\}^w$, and the decoder, Bob, aims to recover x_1, \dots, x_n . Our compression scheme uses a long string which is shared by Alice and Bob but is completely independent of x_1, \dots, x_n . This shared string consists of

- Fixed read/write sequence A and read-only sequence B ;
- A sequence R of k reads where the indices are sampled uniformly at random (i.e., $(\text{read}, a_1), (\text{read}, a_2), \dots, (\text{read}, a_k)$, where $a_1, \dots, a_k \leftarrow [n]$);
- An integer $t \leftarrow [k]$ sampled uniformly at random; and
- A random tape ρ used by \mathcal{DS} .

Since x_1, \dots, x_n are sampled independently and uniformly, their entropy conditioned on the shared string is nw . Therefore, by Shannon’s source coding theorem, the only way for Alice to correctly transmit them to Bob is by sending at least nw bits.

Alice’s encoding:

- Input: nw bits interpreted as $x_1, \dots, x_n \in \{0, 1\}^w$.
- Procedure:
 1. Using ρ and \mathcal{DS} , execute the sequence of requests

$$A, L, B, R,$$

where A, B , and R are taken from the shared string, and $L := (\text{write}, 1, x_1), (\text{write}, 2, x_2), \dots, (\text{write}, n, x_n)$. Define the following collections of cells’ indices that are physically probed during the execution:

- $C_0 := \text{Cells}(L, B \mid A)$. That is, the cells probed during the execution of the L, B sequences.
- $C := C_0 \cap \text{Cells}(R \mid A, L, B)$. That is, the cells probed during the execution of the L, B sequences which are also probed during the execution of the R sequence.

Right after executing A, L, B using ρ , let σ be the local state of \mathcal{DS} , and let $\text{content}(C)$ be the contents of the cells in C .

2. Define $R[1 \dots t - 1] := (\text{read}, a_1), \dots, (\text{read}, a_{t-1})$ to be the sequence of operations that consists of the first $t - 1$ reads from R . For each $i \in [n]$, define $\widehat{R}_{t,i}$ to be a sequence of operations that consists of $R[1 \dots t - 1]$ and then, as its t th operation, it performs a read from index i . That is,

$$\widehat{R}_{t,i} := (\text{read}, a_1), \dots, (\text{read}, a_{t-1}), (\text{read}, i).$$

3. For each $i \in [n]$, using ρ and \mathcal{DS} , execute the sequence of operations

$$A, L, B, \widehat{R}_{t,i}.$$

- We say that $i \in [n]$ (or $\widehat{R}_{t,i}$ correspondingly) is *easy* iff

$$\text{Cells}(\text{(read, } i) \mid A, L, B, R[1 \dots t - 1]) \cap C_0 = \emptyset,$$

and *hard* otherwise. Let $H \subset [n]$ be the set of hard i 's and $h := (x_i)_{i \in H}$ (written in increasing order w.r.t. i).

- For each $i \in [n]$, add i into set H_0 iff \mathcal{DS} answers operation (read, i) *incorrectly* (after the execution of $A, L, B, R[1 \dots t - 1]$). That is, let $i \in H_0$ iff the answer to (read, i) is not x_i . Let $h_0 := (x_i)_{i \in H_0}$ (written in increasing order w.r.t. i).

– Output:

- If $|C| \geq \epsilon k$, output a bit 0, followed by $\text{msg}_0 := (x_1, \dots, x_n)$.
- Else (i.e., $|C| < \epsilon k$), output a bit 1, followed by $\text{msg}_1 := (\sigma, C, \text{content}(C), H, h, H_0, h_0)$.

Bob's decoding:

– Input from Alice is either

- the first bit is 0, followed by $\text{msg}_0 := (x'_1, \dots, x'_n)$, or
- the first bit is 1, followed by $\text{msg}_1 := (\sigma, C, \text{content}(C), H, h, H_0, h_0)$.

– Procedure:

1. If the first bit is 0, output the received x'_1, \dots, x'_n directly. Otherwise, continue as follows.
2. For each hard $i \in [n]$, i.e., $i \in H$, recover x'_i by reading it from h (recall that elements in h are ordered in increasing i).
3. For each incorrect index $i \in H_0$, recover x'_i by reading it from h_0 (recall that elements in h_0 are ordered in increasing i).
4. For each easy and correct $i \in [n]$, i.e., $i \notin H \cup H_0$, recover x'_i using the following steps:
 - (a) Using \mathcal{DS} and randomness ρ , execute the sequence of operations A . Then, replace the content of cells in C with $\text{content}(C)$ and replace the local state of \mathcal{DS} with σ .
 - (b) Using this configuration, randomness ρ , and \mathcal{DS} , execute $\widehat{R}_{t,i}$ and let x'_i be the result of the t th operation in $\widehat{R}_{t,i}$, i.e., (read, i) .

– Output: x'_1, \dots, x'_n .

Correctness of compression. For correctness, we show that Bob always outputs values x'_1, \dots, x'_n such that $x'_i = x_i$ for all $i \in [n]$, where x_1, \dots, x_n are the inputs of Alice. Whenever $|C| \geq \epsilon k$, correctness holds immediately since Alice just sends x_1, \dots, x_n explicitly to Bob. We therefore consider the case where $|C| < \epsilon k$. For every hard $i \in H$ or incorrect $i \in H_0$, we have $x'_i = x_i$ by construction (since it is transmitted explicitly as part of h or h_0). For each easy and

correct $i \in [n]$, executing $\widehat{R}_{t,i}$ (using \mathcal{DS} , local state σ , and random tape ρ) needs only the contents of cells either in C or not in C_0 (observe that $R[1 \dots t - 1]$ needs both and then $\text{easy}(\text{read}, i)$ needs only those not in C_0). Bob can obtain the content of these cells not in C_0 by executing the sequence of operations A . Hence, all the needed information can be obtained by Bob and it is identical to that of Alice. Recall that sequence B is *read-only* so the output is indeed x_i written by L . Therefore, by correctness of \mathcal{DS} (as *writes to and reads from* $i \in [n] \subseteq [N]$ are valid operations), Bob indeed obtains $x'_i = x_i$ for all $i \in [n]$.

5.2 Encoding Size Analysis

We upper bound the expected size of the encoding outputted by Alice. We follow the conventions that i) $|s|$ denotes the *number of bits* of s for any *sequence* s , and ii) $|S|$ denotes the *cardinality* of S for any *set* S .

The encoding consists of a bit j and the message msg_j , where j depends on whether $|C| \geq \epsilon k$. Let **Good** be the indicator for the event that $|C| < \epsilon k$. By the law of total expectation, the expected size is the sum of two cases,

$$\mathbf{E}[|j, \text{msg}_j|] = 1 + \mathbf{E}[|\text{msg}_0| \mid \neg\text{Good}] \cdot \Pr[\neg\text{Good}] + \mathbf{E}[|\text{msg}_1| \mid \text{Good}] \cdot \Pr[\text{Good}].$$

By Eq. (4), we have

$$\Pr[\text{Good}] = 1 - \beta \quad \text{and} \quad \Pr[\neg\text{Good}] = \beta, \tag{5}$$

and by construction, $|\text{msg}_0|$ is always $n\mathbf{w}$ bits. We thus focus on proving an upper bound on the second conditional expectation, namely on $\mathbf{E}[|\text{msg}_1| \mid \text{Good}]$.

Recall that the encoding msg_1 consists of $\sigma, C, \text{content}(C), H, h, H_0, h_0$ and so by linearity of expectation, it suffices to bound the expected size of each component marginally. First, since the local state of \mathcal{DS} is m bits, we know that $|\sigma| \leq m$. Second, by the definition of the event **Good**, we have that

$$\mathbf{E}[|C| \mid \text{Good}] < \epsilon k \quad \text{and} \quad \mathbf{E}[|\text{content}(C)| \mid \text{Good}] < \epsilon k \mathbf{b},$$

where the latter inequality follows since each cell consists of \mathbf{b} bits. Third, for H_0 and h_0 , we have $\mathbf{E}[|H_0|] \leq \delta n$ by δ -correctness of \mathcal{DS} and then linearity of expectation. Hence, we have $\mathbf{E}[|h_0|] \leq \delta n \mathbf{w}$ without conditioning on **Good**. That is, it takes just $\delta n \mathbf{w}$ bits even if Alice had always sent h_0 .

We are therefore left with upper bounding the number of hard read requests $\widehat{R}_{t,i}$, namely, the cardinality of H . For this, we use the fact that the t th read request is online and is made after the previous $t - 1$ requests are executed. That is, after executed $t - 1$ requests where \mathcal{DS} reads cells in C , the set C is fixed. Then, when given the t th request, \mathcal{DS} must touch a new cell not in C (unless it got lucky and it was already in C). Intuitively, this means that \mathcal{DS} must spend probes in order to answer the t th random read request (no matter how many probes were spent on write requests and on previous read requests). Formalizing this intuition into a bound on $|H|$ is done in the following Lemma (see the full version [29] for the proof).

Lemma 2. *Assuming Eq. (4), then $\mathbf{E}[|H| \mid \text{Good}] < (\beta + \epsilon/(1 - \beta))n$.*

Expected size of encoding. We now sum up the expected size of msg_1 sent by Alice conditioned on the case **Good**. Recall that σ takes m bits, and C and $\text{content}(C)$ consume together at most $2\epsilon kb$ bits conditioned on **Good**. The set H can be described simply using a binary string of n bits, where the i th bit indicates whether $i \in H$ or not. To describe h , by Lemma 2, h can be described with at most $(\beta + \epsilon/(1 - \beta))nw$ bits in expectation. The set H_0 is described using n bits as well, but we defer h_0 since its expectation is not conditional. So, the expected size conditioned on **Good** is

$$\begin{aligned} m + 2\epsilon kb + n + (\beta + \epsilon/(1 - \beta)) \cdot nw + n &\leq m + 2enw + n + (\beta + \epsilon/(1 - \beta)) \cdot nw + n \\ &\leq (3\epsilon + \epsilon/(1 - \beta) + 1/8 + \beta) \cdot nw, \end{aligned}$$

where the first inequality follows since $k \leq nw/b$, and the second is since $m \leq enw$ and $w \geq 16$. The total expected size is then

$$\begin{aligned} 1 + \mathbf{E}[|h_0|] + \mathbf{E}[|\text{msg}_0| \mid \neg\text{Good}] \cdot \mathbf{Pr}[\neg\text{Good}] + \mathbf{E}[|\text{msg}_1| \mid \text{Good}] \cdot \mathbf{Pr}[\text{Good}] \\ \leq 1 + \delta nw + nw \cdot \mathbf{Pr}[\neg\text{Good}] + (3\epsilon + \epsilon/(1 - \beta) + 1/8 + \beta) \cdot nw \cdot \mathbf{Pr}[\text{Good}] \\ = 1 + \delta nw + nw\beta + (3\epsilon + \epsilon/(1 - \beta) + 1/8 + \beta) \cdot (1 - \beta) \cdot nw, \end{aligned}$$

where the first term 1 is the bit indicating if the case is **Good** in Alice’s encoding, and the last equality follows since $\mathbf{Pr}[\text{Good}] = 1 - \beta$ (Eq. (5)). Plugging in $\delta = 1/128, \epsilon = 1/128$ and $\beta \leq \alpha = 3/4$, we obtain that the expected encoding size is strictly smaller than

$$1 + (1/128 + 3/4 + (1/4)(1/16 + 1/8 + 3/4)) \cdot nw < nw$$

in bits, where the inequality follows since $n \geq 8$ and $w \geq 16$. By Shannon’s source coding theorem, we thus reached a contradiction which completes the proof of Theorem 5.

6 Separating Offline and Online ORAM

In this section we prove a separation between the offline and online ORAM models. Concretely, we prove the following result.

Theorem 6. *Consider the task of obviously simulating a RAM with N cells each of size $w = \log N$ bits using a RAM of N' cells each of size b bits and using local memory of size m bits for $b, m \in \text{poly} \log N$. There exists an offline ORAM scheme with $N' \in O(N)$ for this task with $o(1)$ I/O efficiency, while every online ORAM scheme for this task must have $\Omega(\log N / \log \log N)$ I/O efficiency (no matter how large N' is).*

Proof. The lower bound follows directly from Theorem 3. Plugging in the values of w, b, m , we get that every *online* ORAM scheme for this task must have $\Omega(\log N / \log \log N)$ I/O efficiency. The upper bound follows from existing results [7, 11] and is deferred to the full version [29].

Acknowledgements. The first author thanks Paul Grubbs for a discussion that motivated him to look into the lower bound of [33] more closely. We also thank Elaine Shi for useful discussions. I. K. is supported in part by an Alon Young Faculty Fellowship and by an ISF grant (No. 1774/20). W.-K. L. is supported in part by a DARPA Brandeis award. This work was done partly while W.-K. L. worked at NTT Research.

References

1. Abraham, I., Fletcher, C.W., Nayak, K., Pinkas, B., Ren, L.: Asymptotically tight bounds for composing ORAM with PIR. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 91–120. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_5
2. Aggarwal, A., Vitter, J.S.: The input/output complexity of sorting and related problems. *Commun. ACM* **31**(9), 1116–1127 (1988)
3. Asharov, G., Komargodski, I., Lin, W.-K., Nayak, K., Peserico, E., Shi, E.: OptORAMa: optimal oblivious RAM. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 403–432. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_14
4. Bindschaedler, V., Naveed, M., Pan, X., Wang, X., Huang, Y.: Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward. In: ACM CCS, pp. 837–849 (2015)
5. Boyle, E., Chung, K.-M., Pass, R.: Large-scale secure computation: multi-party computation for (parallel) RAM programs. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 742–762. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_36
6. Boyle, E., Chung, K.-M., Pass, R.: Oblivious parallel RAM and applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 175–204. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_7
7. Boyle, E., Naor, M.: Is there an oblivious RAM lower bound? In: ITCS (2016)
8. Cash, D., Drucker, A., Hoover, A.: A lower bound for one-round oblivious RAM. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 457–485. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_16
9. Cash, D., K upc u, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. *J. Cryptol.* **30**(1), 22–57 (2017)
10. Chan, T.-H.H., Guo, Y., Lin, W.-K., Shi, E.: Oblivious hashing revisited, and applications to asymptotically efficient ORAM and OPRAM. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 660–690. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_23
11. Chan, T.H., Guo, Y., Lin, W., Shi, E.: Cache-oblivious and data-oblivious sorting and applications. In: SODA, pp. 2201–2220 (2018)
12. Damg ard, I., Meldgaard, S., Nielsen, J.B.: Perfectly secure oblivious RAM without random oracles. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 144–163. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_10
13. Devadas, S., van Dijk, M., Fletcher, C.W., Ren, L., Shi, E., Wichs, D.: Onion ORAM: a constant bandwidth blowup oblivious RAM. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 145–174. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_6
14. Farhadi, A., Hajiaghayi, M., Larsen, K.G., Shi, E.: Lower bounds for external memory integer sorting via network coding. In: STOC (2019)

15. Fletcher, C.W., Dijk, M.V., Devadas, S.: A secure processor architecture for encrypted computation on untrusted programs. In: Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing, pp. 3–8. ACM (2012)
16. Fletcher, C.W., Ren, L., Kwon, A., van Dijk, M., Devadas, S.: Freecursive ORAM: [nearly] free recursion and integrity verification for position-based oblivious RAM. In: ASPLOS (2015)
17. Floyd, R.W.: Permuting information in idealized two-level storage. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 105–109. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_10
18. Fredman, M.L., Saks, M.E.: The cell probe complexity of dynamic data structures. In: STOC. ACM (1989)
19. Gentry, C., Goldman, K.A., Halevi, S., Jutla, C.S., Raykova, M., Wichs, D.: Optimizing ORAM and using it efficiently for secure computation. In: De Cristofaro, E., Wright, M. (eds.) PETS 2013. LNCS, vol. 7981, pp. 1–18. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39077-7_1
20. Gentry, C., Halevi, S., Jutla, C., Raykova, M.: Private database access with HE-over-ORAM architecture. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 172–191. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_9
21. Gentry, C., Halevi, S., Raykova, M., Wichs, D.: Outsourcing private RAM computation. In: FOCS (2014)
22. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. *J. ACM* **43**(3), 431–473 (1996)
23. Goodrich, M.T.: Data-oblivious external-memory algorithms for the compaction, selection, and sorting of outsourced data. In: SPAA (2011)
24. Goodrich, M.T., Mitzenmacher, M.: Privacy-preserving access of outsourced data via oblivious RAM simulation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 576–587. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_46
25. Gordon, S.D., et al.: Secure two-party computation in sublinear (amortized) time. In: CCS (2012)
26. Hubáček, P., Koucký, M., Král, K., Slívová, V.: Stronger lower bounds for online ORAM. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11892, pp. 264–284. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_10
27. Jacob, R., Larsen, K.G., Nielsen, J.B.: Lower bounds for oblivious data structures. In: SODA (2019)
28. Jafargholi, Z., Larsen, K.G., Simkin, M.: Optimal oblivious priority queues. In: SODA (2021)
29. Komargodski, I., Lin, W.K.: A logarithmic lower bound for oblivious RAM (for all parameters). Cryptology ePrint Archive, Report 2020/1132 (2020)
30. Kushilevitz, E., Lu, S., Ostrovsky, R.: On the (in)security of hash-based oblivious RAM and a new balancing scheme. In: SODA (2012)
31. Larsen, K.G.: The cell probe complexity of dynamic range counting. In: STOC (2012)
32. Larsen, K.G., Malkin, T., Weinstein, O., Yeo, K.: Lower bounds for oblivious near-neighbor search. In: SODA (2020)
33. Larsen, K.G., Nielsen, J.B.: Yes, there is an oblivious RAM lower bound! In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 523–542. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_18

34. Larsen, K.G., Simkin, M., Yeo, K.: Lower bounds for multi-server oblivious RAMs. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 486–503. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_17
35. Larsen, K.G., Weinstein, O., Yu, H.: Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. In: 2018 Information Theory and Applications Workshop, ITA, pp. 1–40 (2018)
36. Lin, W., Shi, E., Xie, T.: Can we overcome the $n \log n$ barrier for oblivious sorting? In: SODA (2019)
37. Liu, C., Wang, X.S., Nayak, K., Huang, Y., Shi, E.: OblivM: A programming framework for secure computation. In: IEEE S&P (2015)
38. Lu, S., Ostrovsky, R.: Distributed oblivious RAM for secure two-party computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 377–396. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_22
39. Maas, M., et al.: PHANTOM: practical oblivious computation in a secure processor. In: ACM CCS (2013)
40. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: STOC (1997)
41. Patel, S., Persiano, G., Raykova, M., Yeo, K.: PanORAMa: oblivious RAM with logarithmic overhead. In: FOCS (2018)
42. Patel, S., Persiano, G., Yeo, K.: Lower bounds for encrypted multi-maps and searchable encryption in the leakage cell probe model. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 433–463. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_15
43. Persiano, G., Yeo, K.: Lower bounds for differentially private RAMs. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 404–434. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_14
44. Pătraşcu, M., Demaine, E.D.: Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.* **35**(4), 932–963 (2006)
45. Ren, L., et al.: Constants count: practical improvements to oblivious RAM. In: USENIX Security (2015)
46. Ren, L., Yu, X., Fletcher, C.W., van Dijk, M., Devadas, S.: Design space exploration and optimization of path oblivious RAM in secure processors. In: ISCA (2013)
47. Shi, E.: Path oblivious heap: optimal and practical oblivious priority queue. In: S&P (2020)
48. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_11
49. Stefanov, E., et al.: Path ORAM: an extremely simple oblivious RAM protocol. *J. ACM* **65**(4), 18:1–18:26 (2018)
50. Stefanov, E., et al.: Path ORAM: an extremely simple oblivious RAM protocol. In: CCS (2013)
51. Stefanov, E., Shi, E.: Oblivstore: high performance oblivious cloud storage. In: IEEE S&P (2013)
52. Stefanov, E., Shi, E., Song, D.X.: Towards practical oblivious RAM. In: NDSS (2012)
53. Vitter, J.S.: External memory algorithms and data structures: dealing with massive data. *ACM Comput. Surv.* **33**(2), 209–271 (2001)
54. Wang, X., Chan, T.H., Shi, E.: Circuit ORAM: on tightness of the goldreich-ostrovsky lower bound. In: CCS (2015)

55. Wang, X.S., Huang, Y., Chan, T.H., Shelat, A., Shi, E.: SCORAM: oblivious RAM for secure computation. In: ACM CCS, pp. 191–202 (2014)
56. Wang, X.S., et al.: Oblivious data structures. In: CCS (2014)
57. Weiss, M., Wichs, D.: Is there an oblivious RAM lower bound for online reads? *J. Cryptol.* **34**(3), 18 (2021)
58. Williams, P., Sion, R., Tomescu, A.: Privatefs: A parallel oblivious file system. In: ACM CCS (2012)
59. Yao, A.C.: Should tables be sorted? *J. ACM* **28**(3), 615–628 (1981)
60. Zahur, S., et al.: Revisiting square-root ORAM: efficient random access in multi-party computation. In: IEEE S&P, pp. 218–234 (2016)



Oblivious RAM with *Worst-Case* Logarithmic Overhead

Gilad Asharov¹(✉), Ilan Komargodski^{2,3}, Wei-Kai Lin⁴, and Elaine Shi⁵

¹ Bar-Ilan University, Ramat Gan, Israel
Gilad.Asharov@biu.ac.il

² Hebrew University of Jerusalem, Jerusalem, Israel
ilank@cs.huji.ac.il

³ NTT Research, Sunnyvale, CA, USA

⁴ Cornell University, New York, USA
wklin@cs.cornell.edu

⁵ CMU, Pittsburgh, USA
runting@cs.cmu.edu

Abstract. We present the first Oblivious RAM (ORAM) construction that for N memory blocks supports accesses with *worst-case* $O(\log N)$ overhead for any block size $\Omega(\log N)$ while requiring a client memory of only a constant number of memory blocks. We rely on the existence of one-way functions and guarantee computational security. Our result closes a long line of research on fundamental feasibility results for ORAM constructions as logarithmic overhead is necessary.

The previous best logarithmic overhead construction only guarantees it in an *amortized* sense, i.e., logarithmic overhead is achieved only for long enough access sequences, where some of the individual accesses incur $\Theta(N)$ overhead. The previously best ORAM in terms of *worst-case* overhead achieves $O(\log^2 N / \log \log N)$ overhead.

Technically, we design a novel de-amortization framework for modern ORAM constructions that use the “shuffled inputs” assumption. Our framework significantly departs from all previous de-amortization frameworks, originating from Ostrovsky and Shoup (STOC ’97), that seem to be fundamentally too weak to be applied on modern ORAM constructions.

Keywords: Oblivious RAM · Worst-case overhead · Deamortization

1 Introduction

Imagine a client that wishes to offload a database containing sensitive information to an untrusted server and later access the database and retrieve parts of it. By now, it is well-known that merely encrypting the entries of the database before uploading them to the server does not guarantee privacy (e.g., [6, 21, 22, 41]). Indeed, the access patterns themselves may reveal non-trivial information about the underlying data or program being executed on the data. To mitigate these kinds of attacks, we would like to be able not only to encrypt the underlying data but also to “scramble” the observed access patterns so that they look unrelated

to the data. The algorithmic tool that achieves this goal is called an Oblivious RAM (ORAM).

An ORAM, introduced in the seminal work of Goldreich and Ostrovsky [16, 17], is a (probabilistic) RAM machine whose memory accesses do not reveal anything about the input—including both program and data—on which it is executed. An ORAM construction accomplishes this by permuting data blocks stored on the server and periodic reshuffling them around. Since their introduction more than 30 years ago, ORAMs have also become a central tool in designing various cryptographic systems, including cloud computing design, secure processor design, multi-party computation protocols, and more [4, 12, 13, 15, 26–29, 31, 34, 35, 38–40].

To be useful, ORAMs have to be “efficient”. Whether an ORAM is efficient or not is typically measured by its (asymptotic) *overhead in bandwidth*: that is, how many data items must be accessed in the oblivious simulation as compared to the original non-oblivious implementation. There has been a tremendous effort in designing the most efficient ORAM construction possible [2, 7, 17, 18, 24, 30, 32, 33, 37]. The current record is the OptORAMa scheme by Asharov et al. [2] (building on Patel et al. [30]) who obtained an ORAM with amortized logarithmic overhead. Namely, their ORAM can simulate a RAM of size N so that over the span of T accesses, the total number of accesses would be $O(T \cdot \log N)$. The beautiful lower bound of Larsen and Nielsen [25] (see also [23]) shows that this is essentially the best possible: that is, every ORAM construction must spend *on average* $\Omega(\log N)$ physical accesses per one logical operation.¹

Worst-case overhead. Much of the recent progress on ORAM constructions focuses on reducing its amortized cost [2, 30], whereas the worst-case overhead of an operation was ignored. Specifically, while achieving logarithmic amortized overhead, these constructions have $\Omega(N)$ worst-case overhead, due to the occasional reshuffling operations. This worst-case behavior renders these schemes much less useful in many applications since every now and then an access will “block” until $\Omega(N)$ physical accesses are complete which is clearly unacceptable.

The first to address this problem were Ostrovsky and Shoup [29] who showed how to spread the reshuffling operations over time, and achieve a worst-case $O(\log^3 N)$ overhead version of the original ORAM of Goldreich and Ostrovsky [17]. Related techniques were later applied on other ORAM schemes [7, 19, 24]. In spite of the recent great progress in ORAM constructions, the best known construction in terms of worst-case overhead is from almost a decade ago due to Kushilevitz, Lu, and Ostrovsky [24] who achieved $O(\log^2 N / \log \log N)$ worst-case overhead (and their scheme was further clarified in the subsequent work of Chan et al. [7]). Crucially, the techniques of Ostrovsky and Shoup do not apply

¹ The lower bounds of [23, 25] only apply to “online” ORAMs which support operations that come in an online fashion, one by one. These lower bounds even apply to computationally secure constructions. There is a logarithmic lower bound for “offline” ORAMs which see the whole set of operations ahead of time due to Goldreich and Ostrovsky [17], but it only applies to statistically secure constructions in the balls-and-bins model (see Boyle and Naor [5]).

to the recent constructions that are based on “randomness reusing” of [2,30], as we elaborate below in Sect. 2.

Thus, the current state of affairs leaves open the following fundamental question (also raised in [2]):

Is there a worst-case logarithmic overhead ORAM? That is, is there an ORAM construction that can simulate every logical operation with $O(\log N)$ physical accesses?

1.1 Our Contributions

Optimal worst-case overhead ORAM. We propose a new ORAM construction that achieve logarithmic worst-case overhead (in the memory size), while consuming $O(1)$ client-side storage, and $O(N)$ server-side storage. Here, N denoted the memory size. Obliviousness of our construction relies on the existence of one-way functions. Our result answers an important question left open by the recent OptORAMa work [2].

Theorem 1.1. *There is a computationally-secure ORAM with $O(\log N)$ worst-case overhead assuming that one-way functions exist.*

The construction that achieves Theorem 1.1 is in the most standard model and make the same set of assumptions as all prior computationally-secure ORAM schemes. We assume a standard word-RAM where each memory word has at least $w = \log N$ bits, i.e., large enough to store its own logical address. We assume that word-level addition and boolean operations can be done in unit cost. We assume that the CPU has constant number of private registers. We additionally assume that a single evaluation of a PRF resulting in at least word-size number of pseudo-random bits, can be done in unit cost.

Technically, we significantly depart from all previous “deamortized” ORAM constructions. While all previous works² almost directly use the approach of Ostrovsky and Shoup [29], it seems like this approach is fundamentally too weak to be applied on modern ORAM constructions that achieve amortized logarithmic overhead [2,30]. To this end, we build a new set of novel algorithmic tools from ground up and show how to amend the construction of Asharov et al. [2] so that it could be deamortized.

Linear-time oblivious deduplication. A noteworthy building block that we develop is an algorithm for efficient oblivious *deduplication*. Consider two sets of n elements A and B , where the goal is to obliviously compute $A \cup B$, that is, merge them into one larger set while removing duplicate elements. (Note that we assume that the elements within A are distinct, and ditto for B . Also, we assume that if there is a duplication, we keep the copy coming from A for concreteness.) Oblivious deduplication is a central building block in many previous worst-case

² Here we ignore tree-based constructions [32,33,37] since it is not known how to use them to get even amortized logarithmic overhead.

efficient ORAM constructions. Before this work, the only known solution was to apply a generic oblivious sort on the concatenation of A and B , followed by a linear scan which “deletes” the duplicates (after the sort, duplicates are adjacent in memory). Using the best known oblivious sort (e.g., AKS [1]), this approach would incur $O(n \cdot \log n)$ time. Eventually the extra log factor propagates into the overhead of the ORAM, resulting in $O(\log^2 N / \log \log N)$ worst-case overhead (at best).

We show that the extra $\log n$ overhead can be avoided by designing a *linear time* algorithm for this task which is oblivious if the input arrays are randomly shuffled. That is, we show that if A and B are shuffled with independent secret permutations, then there is a way to compute $A \cup B$ in time $O(n)$ while maintaining obliviousness. Linear overhead is clearly the best possible (since just reading the input takes linear time), matching the state of the art without obliviousness.

Theorem 1.2. *There is a linear time (probabilistic) algorithm that gets as input two sets A and B and outputs $A \cup B$. The algorithm is further (computationally) oblivious if A and B are independently secretly shuffled and if one-way functions exist.*

Conclusions and Open Problems. We see our work as closing a long line of research on theoretical feasibility results for ORAM constructions. Our *worst-case logarithmic overhead ORAM* result, at least asymptotically, is optimal in terms of computational overhead. Unfortunately, the concrete constant, inherited from [2], underlying our construction is rather large. Using better generic building blocks (say the oblivious compaction algorithm of Dittmer and Ostrovsky [11]) we can get a much better constant but we believe that it is still too large for deployment. Whether there is a construction with (worst-case) logarithmic asymptotic overhead and a *small* constant is a major open problem.

Another exciting open problem is to bring down the cost of *statistically secure* ORAMs closer to $O(\log N)$ or prove that it is impossible. Specifically, the best statistically secure ORAMs has overhead $O(\log^2 N / \log \log N)$ [9] and it relies on the tree-based paradigm due to Shi et al. [32] which was later improved by [9, 10, 13, 33, 37]. Interestingly, tree based ORAMs have so far been more concretely efficient than hierarchical ones and this question is also somewhat related to the previous one.

Lastly, we mention a recent work of Asharov et al. [3] who gives an optimal Oblivious *PRAM* (OPRAM). An OPRAM is an extension of ORAM to the parallel setting where several processors make concurrent accesses to a shared memory. Their main result is that (assuming one-way functions) any PRAM with memory capacity N can be obliviously simulated in space $O(N)$, incurring only *amortized* $O(\log N)$ overhead in work and (worst-case) $O(\log N)$ overhead in depth. We believe that our techniques can be further extended to obtain a *worst-case* logarithmic work and depth overhead OPRAM, but this is left for future work.

2 Technical Overview

2.1 Background: Underlying ORAM Without Deamortization

The hierarchical paradigm and oblivious hash tables. We will build from an underlying (amortized) ORAM scheme which follows the hierarchical paradigm established by Goldreich and Ostrovsky [16,17]. An ORAM scheme in the hierarchical paradigm can be viewed as a technique to reduce the task of constructing ORAM to the task of constructing an oblivious hash table. Specifically, A hierarchical ORAM typically consists of $\log_2 N + 1$ levels numbered $0, 1, \dots, n$. Each level i is an *oblivious hash table* that can contain at most 2^i elements. An oblivious hash table is a data structure that supports the following operations:

- **Build** takes an input array containing (key, value) pairs and creates the data structure (we also say a pair is an element, a block, or an item);
- **Lookup** receives a key k , and returns the value corresponding to the key k contained in the data structure, or returns \perp if not found or if the key looked up is dummy (denoted \perp).
- **Extract** is called when the data structure is destructed, and returns a list of unvisited items in the data structure.

Almost all known ORAM schemes in the hierarchical paradigm guarantee the following *non-recurrent* invariant: for each oblivious hash table in the hierarchy, the same real (i.e., non-dummy) key must be looked up at most once during the life-cycle of the data structure. Therefore, the data structure only needs to provide obliviousness if this non-recurrent assumption is respected. Finally, an oblivious hash table often has an access budget in the sense that it can only support up to an a-priori fixed number of lookup requests. Typically this budget is at least n which is the size of the array input into **Build**.

Achieving amortized logarithmic overhead. The original oblivious hash table implementation suggested by Goldreich and Ostrovsky [16,17] is slow and takes $O(n \log n)$ time to build for an input array of size n . This would result in a non-optimal ORAM scheme. Instead, we adopt the efficient oblivious hash table suggested by Asharov et al. [2] (which is built upon Patel et al. [30]). Asharov et al. showed an oblivious hash table with $O(n)$ build time and $O(1)$ lookup overhead, except with the following input assumptions, output requirement, and caveat:

- *Randomly shuffled requirement:* the input array of **Build** must be randomly shuffled; moreover, the **Extract** function outputs the unvisited blocks in a random order. In either case, the randomness is hidden from the adversary.
- *Size assumption:* to obtain negligible in λ failure probability, the construction only works for hash tables that are at least $\text{poly} \log(\lambda)$ in size.
- *Stash:* while the main hash table data structure can indeed be looked up in constant time, the hash table construction actually comes with a stash, and sometimes the element to be looked up actually resides in the stash. Each stash has expected constant size but with noticeable probability, the size can be as large as $O(\log \lambda)$.

We briefly overview how past works [2,30] deal with the above imperfectness to make the ORAM scheme work. The recent work by Patel et al. [30] and Asharov et al. [2] show that by relying on the “residual randomness”, ORAM constructions can respect the randomly shuffled input assumption. Specifically, when each oblivious hash table is destructed in the ORAM, the unvisited elements in the hash table appear in a random order, and the random permutation is hidden from the adversary. The size assumption can be dealt with by using yet another designated, slower, data-structure for smaller levels in the ORAM that are less than $\text{poly log}(\lambda)$ in size. Finally, the stash issue can be dealt with by merging the stashes of all oblivious hash tables into a single one, and accessing the merged stash once and for all for each ORAM request—one can prove that the merged stash is at most poly logarithmic in size except with negligible probability, and is stored in a designated data structure to allow fast lookup (i.e., taking strictly logarithmic time).

Simplifying assumptions. For ease of understanding, let us first ignore the size assumption and the stash issue mentioned above—these introduce additional technicalities for constructing an optimal, deamortized ORAM as we shall mention shortly. For the time being, we pretend that we can indeed have an oblivious hash table for randomly shuffled inputs can be built in linear time, regardless of the size, and moreover, assuming that lookup need not deal with the stash technicality.

Underlying ORAM scheme without deamortization. With these simplifying assumptions, we can construct an ORAM scheme as follows—our description below matches the rebuild description of Asharov et al. [3] in their optimal OPRAM scheme, and is a variant of Goldreich and Ostrovsky’s original hierarchical construction.

Assume that the total memory size N is a power of 2. Imagine that there are $\log_2 N + 1$ levels, where the i -th level is an oblivious hash table (for randomly shuffled inputs) of capacity 2^i . We use T_0, \dots, T_L to denote all the $L + 1$ levels where $L = \log_2 N$.

In the steady state of the ORAM (i.e., ignoring the initial time steps when the levels are not yet populated), every level except level 0 is either *half full* (HF) or *full* (F). A level $i > 0$ is *half full* iff it contains up to 2^{i-1} real blocks. A level i is *full* iff it may contain up to 2^i real blocks³. Level 0 is either empty or full, and as we shall see, it is guaranteed to be empty at the end of every ORAM request. Whenever a new ORAM request arrives asking for the block at logical address addr , we do the following:

- *Fetch phase.* From $i = 0$ to L , we look up each oblivious hash table for the logical address addr ; once the block is found, for all subsequent levels, we instead look for a dummy block.

³ The actual number of real blocks may be smaller if the requests keep asking for the same block or a small set of blocks. The maximum load is achieved when the ORAM requests cycle through addresses $1, 2, \dots, N$ in a round-robin fashion.

– *Maintain phase.* The block at `addr` just fetched is updated if necessary, and then it is entered into the smallest level (i.e., level 0), which makes the smallest level full. At this moment, let ℓ be either the smallest level that is half full or $\ell = L$ if all levels are full. Regardless of which case, all levels $0, 1, \dots, \ell - 1$ are full. We now perform the following rebuild procedure:

1. For each level $i = 0$ to $\ell - 2$ in parallel:
 - let $T'_{i+1} := \text{Build}(\text{Intersperse}(T_i.\text{Extract}(), D_i))$ —where D_i is an array of size 2^i containing only dummy elements, and `Intersperse` merges two randomly shuffled arrays into a randomly shuffled array.
2. Let $T'_\ell := \text{Build}(T_{\ell-1}.\text{Extract}() \cup T_\ell.\text{Extract}())$ while we also remove dummy elements when unifying the two arrays;
3. Replace T_1, \dots, T_ℓ with the new hash tables T'_1, \dots, T'_ℓ and let T_0 be emptied.

After this rebuild procedure, $T_0, \dots, T_{\ell-1}$ are all half full (and in fact T_0 is empty), and T_ℓ becomes full. In the above Steps 1 and 2, one can also imagine that each level $1, \dots, \ell - 1$ is “*rebuilding itself down*” into the next level (and we will use this terminology later). For ease of understanding, the maintain phase is depicted in Fig. 1.

Fact 2.1. *In the above construction, a level $i \geq 1$ switches state (either from half full to full, or vice versa) every 2^{i-1} requests. Similarly, a level i wants to rebuild itself down every 2^i requests—this also coincides with when level $i + 1$ refreshes.*

Assuming that the oblivious hash table supports `Build` in linear-time (for randomly shuffled inputs), supports `Lookup` in constant time, and moreover, its `Extract` function outputs unvisited blocks in a random order, then the above ORAM scheme is secure and achieves $O(\log N)$ amortized overhead.

2.2 Why Existing Deamortization Techniques Fail

Clearly the scheme mentioned in Sect. 2.1 cannot give a worst-case efficient ORAM. For instance, when rebuilding the largest level (which happens every N accesses), just reading it requires $O(N)$ work. We describe existing deamortization techniques and explain why they are incompatible with the new generation of amortized optimal ORAM constructions.

Ostrovsky and Shoup [29] proposed a deamortization technique that, roughly speaking, “spread” the rebuild procedures over many accesses rather than performing them atomically. The challenge is how to support accesses to the level while it is being rebuilt.

To do this, first, we modify the access process so that it does *not* delete an element once it is found in some level (while it is still reinserted into the smallest level). With this change, it is immediate that except for the smallest level, the only time the contents of a level changes, is during the rebuild procedure and in the latter we always just “pull” content from its previous (i.e., smaller) level. That is, it takes 2^i accesses until the content of the i th table is modified, and

it will be filled with the content of table $i - 1$ which is also fixed and known. Thus, the rebuild process for a level can be done slowly and in advance across many accesses as follows: each level has a table called `CURRENTACTIVE` and has another table that is `UNDERCONSTRUCTION`, which is being slowly built from `CURRENTACTIVE` and the previous level `CURRENTACTIVE`. When the construction of the level is complete, we just update the pointer of `CURRENTACTIVE` to the new rebuilt table, and start a new `UNDERCONSTRUCTION` version, thus doing the rebuild in the deamortized sense, by spreading the cost uniformly.

There is one very important technical detail with the above approach: since we never actually delete elements, the same key may (and will) appear multiple times in the structure, perhaps with different values. The important invariant is that the *newest* version of the element is always the one that resides in the smaller level. At some point, these copies will meet in the same level during some rebuild process and then the older copy will be suppressed and discarded. The later task is known as oblivious **deduplication** and it is usually implemented using oblivious sort.

Multiple variants and adaptations of the above deamortization technique were used in previous ORAM constructions [7, 19, 24]. However, as we shall argue next, there are inherent obstacles one runs into while trying to apply it on the more recent (amortized) logarithmic overhead ORAM constructions [2, 30].

Challenge I: Access-while-rebuild breaks security. Recall that in the deamortization technique of Ostrovsky and Shoup, we start rebuilding the table into the next level while we also access it at the same time. However, when applied on the ORAM of [2, 30], this completely breaks the input assumption (and therefore security) and is not compatible with the “residual randomness” technique: A lookup that is performed while building reveals the position of an element in the new table—the security of the latter inherently relies on the permutation being completely secret.

To elaborate further, in Ostrovsky and Shoup we know in advance the content of the levels and we start the rebuild process ahead of time, while still allowing accesses to the same levels. In the context of in Ostrovsky and Shoup, this is *secure* as we re-randomize the levels (by obviously sorting/shuffling it) during the rebuild process. However, in our case we cannot re-randomize the levels as this is too expensive, and we follow the residual randomness technique. Moreover, we cannot know in advance which elements will be looked up in the previous level, i.e., the residual randomness will be consumed and thus for security we are not allowed to reuse randomness.

Challenge II: Deduplication takes quasi-linear time. As mentioned, multiple copies of the same key will appear during the lifetime of the ORAM (some might be with different values) and so a deduplication mechanism is needed. More precisely, the task is to compute $A \cup B$ given two sets A and B of size n . (Assume we prefer the copy in A over B for concreteness.) The best known algorithm uses oblivious sort and takes quasi-linear time. However, if we want to have a logarithmic overhead ORAM construction we must implement it in *linear*

time. Note that, even ignoring obliviousness, it is not immediately obvious how to do this in linear time. The most natural approach is to use sorting, but better algorithms can be achieved using hashing tools.

Challenge III: Cannot “lock” shared memory. As mentioned, each level in [2, 30] *effectively* supports lookup in constant time, but this is due to the “shared stash” technique (previously used in [9, 18, 20, 24]). Specifically, in isolation, each level requires $O(1)$ accesses to a main table and an additional scan of a $O(\log N)$ -size stash. The shared stash trick utilizes the fact that there are many levels and an access will translate to multiple lookups for the same key in many levels—this allows us to merge all of the stashes into a global one and scan it only once for all levels. Therefore, the number of accesses per level is $O(1)$ *amortized*. This makes the ORAM construction not completely black-box in a hash table and therefore less compatible with Ostrovsky-Shoup [29]. Concretely, while we rebuild a table, we *cannot* “lock” the global stash as we need to allow accesses to it (addition and removal of elements) while handling other ORAM accesses.

2.3 Our Deamortization Approach

We now intuitively describe how to deamortize the ORAM scheme mentioned in Sect. 2.1. To address Challenge I mentioned in Sect. 2.2, whenever some level is involved in a rebuild, i.e., its destructor `Extract` function is being called, this level no longer can support lookups. Yet the ORAM must continue to serve requests. We propose a new pipelining approach that works with this new constraint that comes with the new amortized logarithmic-overhead ORAMs.

Our key idea is to maintain two copies of each level, henceforth called the `A-copy` and the `B-copy` respectively, each uses its own *independent* randomness. At a high level, whenever some level in the `A-copy` involved in a rebuild and being destructed, the *corresponding* level in the `B-copy` fills in the lookups; and whenever some level in the `B-copy` is involved in a rebuild and being destructed, the *next level* in the `A-copy` can fill in. This guarantees that we never lookup and rebuild from a table at the same time – while rebuilding all lookups are performed at a different copy that uses independent randomness. That is, we essentially split the queries between `A` and `B`, so that we can reuse randomness in each hierarchy by itself. Whenever an element is found in one copy, we update its content and put it in the top of the two hierarchies of `A` and `B`.

We next elaborate our idea in the following schedule of deamortization. Henceforth, we use A_i and B_i to denote the i -th level in the `A-copy` and `B-copy`, respectively. In our earlier non-deamortized scheme, recall that a level $i \geq 1$ is reconstructed every 2^{i-1} requests. We may imagine that there is some counter `ctr` that increments upon every request, and thus a level $i \geq 1$ is refreshed whenever the `ctr` = $k \cdot 2^{i-1}$ for some integer k . For simplicity, in this overview we ignore the treatment of the first level and the last level, and just look at intermediate levels. To achieve deamortization, the idea is to rely on careful pipelining to maintain the following schedule:

- The level A_i is refreshed (i.e., completes reconstruction) at time step $\text{ctr}_1 = k \cdot 2^i + 2^{i-2}$ where k is an integer. Level A_i is now “half full” and just pulled new content from level $i - 1$.
 - The level B_i is refreshed at time $\text{ctr}_2 = k \cdot 2^i + 2 \cdot 2^{i-2}$, i.e., just a little later than the corresponding A_i finished reconstruction.
 - At time $\text{ctr}_2 = k \cdot 2^i + 2 \cdot 2^{i-2}$, level A_i starts pulling more content from A_{i-1} . This will take 2^{i-2} time, which will finish at $\text{ctr}_3 = k \cdot 2^i + 3 \cdot 2^{i-2}$. During this rebuild period $[\text{ctr}_2, \text{ctr}_3]$, A_i is dysfunctional and cannot support accesses; fortunately, B_i contains identical contents as A_i (but randomized differently for obliviousness), and therefore B_i can fill in for the lookups.
 - At time ctr_3 , A_i finishes, and B_i can catch up and pull information from level B_{i-1} . It will start rebuilding and will finish at time $\text{ctr}_4 := \text{ctr}_3 + 2^{i-2}$. At this time range B_i is dysfunctional, however, its content is also in A_i , which also hold in addition some fresher content from A_{i-1} .
 - After B_i finishes and becomes full, A_i wants to push all its content down into level A_{i+1} while also pulling new content from level A_{i-1} . However, the rebuild of level A_{i+1} takes twice as much time. Thus, level A_i will have new content already at time $\text{ctr}_5 := \text{ctr}_4 + 2^{i-2}$, while level A_{i+1} will have the old content of A_i only at time $\text{ctr}_6 := \text{ctr}_5 + 2^{i-2}$. Luckily, the content is not lost; The content exists all this time at the B_i copy, which is activated and functional in $[\text{ctr}_4, \text{ctr}_6]$.
- At time ctr_5 , we essentially finished a cycle, i.e., A_i is half full and just finished refreshing, and we are essentially back to the first item.

Recall that rebuilding A_i (or B_i , resp.) from A_{i-1} (or B_{i-1} , resp.) takes work $\Theta(2^i)$. This amount of work is spread across $\text{ctr}_2 - \text{ctr}_1$ (or $\text{ctr}_3 - \text{ctr}_2$, resp.) time steps. One can verify that indeed, $\text{ctr}_2 - \text{ctr}_1 = \Theta(2^i)$ and thus in each time step, a constant amount of work is performed associated with the rebuilding of A_i (or B_i , resp.). At most $O(\log N)$ many levels are being rebuilt simultaneously, and thus the total amount of work per time step associated with rebuilds is $O(\log N)$. Observe also that the time in between two adjacent rebuilds of A_i is only 2^{i-1} , and the hash table A_i can support up to 2^i requests, so we will not run out of the access budget. A similar observation holds for B_i .

In our final scheme, we will actually have different designated place for the table A_i when it is half-full, and a different placed for A_i when it is full, denoted as A_i^{HF} and A_i^{F} , respectively (likewise for B_i^{HF} and B_i^{F}). Thus, we have 4 tables at the same level, but (the newest version of) each element has exactly 1 copy in each of A_i and B_i . Moreover, only one copy of $(A_i^{\text{HF}}, A_i^{\text{F}})$ is valid at any given time, likewise for B_i . We can view A_i^{HF} and A_i^{F} as two possible states of A_i , and then we have just two copies in each level, or as independent hash tables, and then we have four tables. The rebuild schedule is depicted in Fig. 3.

Why deduplication is necessary in the deamortized scheme. The rebuild procedure of the deamortized scheme is otherwise the same as the underlying non-deamortized scheme except that a deduplication pre-processing step is necessary whenever we are building a half full level A_i and a full level A_{i-1} to create a

new level A_i (and the same for the B-copy). This is because in the deamortized scheme, the rebuilding process is happening while the ORAM is still serving queries. For example, A_i may be rebuilding itself down into A_{i+1} , and the corresponding B_i is taking over as A_i in serving queries. During the rebuild, a memory block at address addr^* may get rebuilt from A_i , into A_{i+1} , but it can also be requested during the same rebuild interim, causing a separate copy of addr^* to be entered into the smallest level. Duplicates will be suppressed when two duplicate copies of the same addr^* “meet” in a future rebuild, while lookup guarantees that the freshest copy (which resides in the level that is smallest compared to any other copy) will be found.

We next describe how to accomplish oblivious deduplication in linear time.

2.4 Linear-Time Oblivious Deduplication

To address Challenge II from Sect. 2.2, another contribution of our paper is a linear-time oblivious deduplication algorithm that takes advantage of the fact that the input arrays to be deduplicated are randomly shuffled. To see the main idea, let us first describe a *non-oblivious* algorithm that runs in linear time. (This is already not trivial.) While the most natural implementation is to sort the concatenated array and then perform a linear scan while removing duplicates (which reside in adjacent positions after the sort), the cost of this implementation is dominated by the sort. Using the best sorting algorithms in the RAM model, one can achieve $o(n \cdot \log n)$ cost [14, 36] but we still do not know of linear time sorting algorithms.

Our idea to get a linear time algorithm is to use hashing tools. Specifically, consider a Cuckoo hash—this is a hash table that supports lookup by just 2 accesses to a main table and another scan of say $O(\log \lambda)$ size stash. (The stash needs to be of this size to guarantee the probability of failure is negligible in λ .) Consider hashing X_1 and X_2 independently into a Cuckoo hash tables T_1, T_2 each having a long enough stash so that the hashing succeeds with all but negligible probability. For $i \in \{1, 2\}$, denote $T_i = (T_i^M, T_i^S)$ where T_i^M is the main table of T_i and T_i^S is the stash of T_i . This costs just $O(n)$. Now, we can perform the deduplication as follows:

1. For each element in the stash of T_1 , i.e., T_1^S , we perform a full lookup in T_2 . That is, for each of the $O(\log \lambda)$ elements in T_1^S , we touch $O(1)$ elements in T_2^M and then scan T_2^S . Doing so, we remove the elements from T_2 that are also in T_1^S . This costs overall $O(\log^2 \lambda)$.
2. So far, we took care of elements that appear in T_2 and in T_1^S and we need to remove duplicates which appear in T_2 and T_1^M . For this, we scan every remaining element in T_2 and for each we touch the two locations in T_1^M to check if it is there. If so, we delete it from T_2 . Crucially, now we do not need to visit the stash T_1^S . When we look for the elements in $T_2 \setminus T_1^S$, we can go directly to the main table and spend $O(1)$ work per lookup, as we know that the element is not in the stash T_1^S !

Eventually, we concatenate the elements of T_1 together with what is left in T_2 after performing the above process. Overall, the cost of this algorithm is $O(n + \log^2 \lambda) = O(n)$ whenever n is large enough compared to λ .

This algorithm is clearly non-oblivious and “naively” replacing the Cuckoo hash with an oblivious version thereof does not meet our goal since known constructions require $\omega(n)$ time for building (since building an oblivious Cuckoo hash uses oblivious sort). This is where the “shuffled inputs” assumptions comes into play: we do not necessarily need the full power of Cuckoo hashing since we are guaranteed that the input lists are shuffled. Therefore, instead of using Cuckoo hash, we use *in a white-box manner* the linear-time oblivious hash table for shuffled inputs from [2] (see also Sect. 3.4). This hash table has linear build time and is secure only when the input array is randomly shuffled; otherwise, it behaves conceptually in a similar manner to “standard” oblivious Cuckoo hash: lookup is performed by a scan of a stash and $O(1)$ accesses to a “main table”. We therefore manage to use it for our purpose, deduplication.

Dealing with the shared stash. So far, we have assumed an idealized oblivious hash table (for randomly shuffled inputs) without stashes. As mentioned earlier, known instantiations of the oblivious hash table with the desired efficiency requirements have an extra stash that must be visited during a `Lookup` operation, and the constant-time lookup is only possible with a “shared stash” trick, i.e., by merging all logarithmically many stashes into a globally shared one.

Accommodating the shared stashes creates extra technicalities, as we mentioned in Challenge III in Sect. 2.2. To deal with them, the main idea is to support more fine-grained access to the shared memory area. That is, while in [2], every `extract` requires to atomically scan the shared memory to retrieve all elements that “belong” to some given level, in our construction we avoid such linear scans by using more efficient data structures. Specifically, we use a version of an oblivious dictionary which supports lookup of elements w.r.t. various auxiliary keys such as the level they came from or the logical address.

To elaborate, recall that each level is associated with a stash of $O(\log \lambda)$ elements, and thus the shared stash consists of $O(\log N \cdot \log \lambda)$ elements. All such elements store the oblivious dictionary accompanied with the auxiliary keys. Specifically, an element is inserted to the dictionary when a level is newly built, is queried by a logical address when lookups are performed during the `fetch` phase, and is popped from the dictionary when a level is being extracted. Recall that each operation of the oblivious dictionary takes only $\text{poly} \log(\log N + \log \lambda)$ time, e.g., by instantiating a perfect ORAM [8], and that each level is at least $\text{poly}(\log N + \log \lambda)$ in size. The efficiency follows as inserting or popping elements take only a $o(1)$ fraction of time during build or extract a level, and only $O(1)$ queries are performed during a `fetch`. The security follows since the dictionary is perfectly oblivious.

Organization. The remaining of the paper is organized as follows. In Sect. 3 we provide the preliminaries, which includes definition of obliviousness, some basic building blocks we use in our construction, our 2-key dictionary and revisit and

overview the oblivious hash table construction of [2]. In Sect. 4 we provide our deduplication algorithm, and in Sect. 5 we provide our deamortized construction. The case of combining the stashes is deferred to the full version.

3 Preliminaries

The security parameter is denoted λ and it is given as input to algorithms in unary (i.e., as 1^λ). A function $\text{negl} : \mathbb{N} \rightarrow \mathbb{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\text{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Two sequences of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *computationally indistinguishable* if for any probabilistic polynomial time algorithm \mathcal{A} , there exists a negligible function $\text{negl}(\cdot)$ such that $|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]| \leq \text{negl}(\lambda)$ for all $\lambda \in \mathbb{N}$. For $n \in \mathbb{N}$, denote $[n] = \{1, \dots, n\}$.

Random-access machines (RAM). A RAM (or a RAM program) is an interactive Turing machine that consists of a memory and a CPU. The program maps some input to an output where its computation is performed by the CPU and using the interaction with the memory. The memory is denoted as $\text{mem}[N, w]$ and is indexed by the logical address space $[N] = \{1, 2, \dots, N\}$. We denote by w to denote the bit-length of each block. The CPU has an internal state that consists of $O(1)$ words. The memory supports read/write instructions ($\text{op}, \text{addr}, \text{data}$) where $\text{op} \in \{\text{read}, \text{write}\}$, $\text{addr} \in [N]$ and $\text{data} \in \{0, 1\}^w \cup \{\perp\}$:

- If $\text{op} = \text{read}$ then $\text{data} = \perp$ and the returned value is the content of the block located in the logical address addr in the memory.
- If $\text{op} = \text{write}$ then the memory data in logical address addr is updated to data .

We use the standard setting that $w = \Theta(\log N)$ (so an address can be stored in a word). We follow the standard convention that the CPU performs one word-level operation per unit time, i.e., such that additions or subtraction (arithmetic operations), bitwise operations such as AND, OR, NOT or shift, memory accesses or evaluation of pseudorandom function.

3.1 Oblivious Machines

Intuitively, we say that a machine M is *oblivious* if there exists a simulator that can simulate its access pattern without knowing the input. Specifically, there exists a simulator Sim such that, for all inputs x , all memory accesses in the computation $M(x)$ can be simulated by Sim where Sim just receives the length of x (i.e., $|x|$) and not the input itself.

We say that a RAM program M_f oblivious simulates a (deterministic) RAM program f if for every input x it holds that $M_f(x) = f(x)$, and that M_f is oblivious. For randomized functionalities, we require that the joint distribution of the output of M_f and the access pattern of the simulator is indistinguishable from

the joint distribution of the output of f and its access pattern in the computation. See discussion in [2]. We are now ready for the definition of oblivious simulation:

Definition 3.1 (Oblivious simulation). *Let $f, M_f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be two RAM machines. We say that M_f obliviously simulates f if there exists a probabilistic polynomial time simulator Sim such that for every input $x \in \{0, 1\}^*$, the following holds:*

$$\{(\text{out}, \text{Addr}) : (\text{out}, \text{Addrs}) \leftarrow M_f(1^\lambda, x)\}_\lambda \approx \left\{ \left(f(x), \text{Sim}(1^\lambda, 1^{|x|}) \right) \right\}_\lambda$$

depending on whether \approx refers to computational, statistical, or perfectly indistinguishable we say that M_f is computationally, statistically, or perfectly oblivious, respectively.

Reactive random-access machines. We consider functionalities that are reactive, i.e., proceed in stages, where the functionality preserves an internal state between stages. Such a reactive functionality can be described as a sequence of RAM machines, where each machine also receives as an input a state, updates it, and the output is the input state for the next machine. We use it to capture building blocks such as oblivious hash tables (see Sect. 3.4).

A reactive machine \mathcal{F} receives commands of the form $(\text{command}_i, \text{inp}_i)$ and produces an output out_i while maintaining some (secret) internal state. Our definition considers an adversary \mathcal{A} (a distinguisher) that participates in either a real execution or an ideal one, and with each command receives the access pattern (resp. simulated access pattern) and the output of the algorithm (resp. output of the functionality). The adversary \mathcal{A} can then adaptively choose the next command to execute.

Definition 3.2 (Oblivious simulation of a reactive functionality). *We say that a reactive machine $M_{\mathcal{F}}$ is an oblivious implementation of the reactive functionality \mathcal{F} if there exists a PPT simulator Sim such that for any non-uniform PPT (stateful) adversary \mathcal{A} , the view of the adversary \mathcal{A} in the following two experiments $\text{Expt}^{\text{real}, M_{\mathcal{F}}}(1^\lambda)$ and $\text{Expt}^{\text{ideal}, \mathcal{F}}_{\mathcal{A}, \text{Sim}}(1^\lambda)$ is computationally indistinguishable:*

$\text{Expt}_{\mathcal{A}}^{\text{real}, M_{\mathcal{F}}}(1^\lambda):$ <p>Let $(\text{cmd}_i, \text{inp}_i) \leftarrow \mathcal{A}(1^\lambda)$ Loop while $\text{cmd}_i \neq \perp$: $\text{out}_i, \text{Addr}_i \leftarrow M_{\mathcal{F}}(1^\lambda, \text{cmd}_i, \text{inp}_i)$ $(\text{cmd}_i, \text{inp}_i) \leftarrow \mathcal{A}(1^\lambda, \text{out}_i, \text{Addr}_i)$</p>	$\text{Expt}_{\mathcal{A}, \text{Sim}}^{\text{ideal}, \mathcal{F}}(1^\lambda):$ <p>Let $(\text{cmd}_i, \text{inp}_i) \leftarrow \mathcal{A}(1^\lambda)$ Loop while $\text{cmd}_i \neq \perp$: $\text{out}_i \leftarrow \mathcal{F}(\text{cmd}_i, \text{inp}_i)$. $\text{Addr}_i \leftarrow \text{Sim}(1^\lambda, \text{cmd}_i)$. $(\text{cmd}_i, \text{inp}_i) \leftarrow \mathcal{A}(1^\lambda, \text{out}_i, \text{Addr}_i)$</p>
---	---

We define statistical or perfect simulation analogously, requiring the two experiments to be either statistically close or identically distributed.

ORAM simulation overhead. We consider the standard ORAM functionality, implementing a logical memory. In this functionality, the user gets to choose the next command (i.e., either read or write) as well as the address and data according

to the access pattern it has observed so far. During its life span, the functionality holds (as an internal state) N memory blocks, each of size w . Denote the internal state $\mathbf{X}[1, \dots, N]$. Initially, $\mathbf{X}[\text{addr}] = 0$ for every $\text{addr} \in [N]$. The functionality is as follows:

- **Access**($\text{op}, \text{addr}, \text{data}$): where $\text{op} \in \{\text{read}, \text{write}\}$, $\text{addr} \in [N]$ and $\text{data} \in \{0, 1\}^w$.
1. If $\text{op} = \text{read}$, set $\text{data}^* := \mathbf{X}[\text{addr}]$.
 2. If $\text{op} = \text{write}$, set $\mathbf{X}[\text{addr}] := \text{data}$ and $\text{data}^* := \text{data}$.
 3. Output data^* .

Typically, the metric of interest for ORAM construction is known as *computation overhead* and it is defined as the (multiplicative) blowup in runtime of the compiled program. We distinguish between the worst-case and the amortized variants:

- **Amortized computation overhead:** We say that the amortized computation overhead of the ORAM is $g: \mathbb{N} \rightarrow \mathbb{N}$ if for every sequence of operations $((\text{op}_1, \text{addr}_1, \text{data}_1), \dots, (\text{op}_q, \text{addr}_q, \text{data}_q))$ at most $g(N) \cdot q$ computation steps are taken during the execution of the ORAM.
- **Worst-case computation overhead:** We say that the amortized computation overhead of the ORAM is $g: \mathbb{N} \rightarrow \mathbb{N}$ if every sequence of operations $((\text{op}_1, \text{addr}_1, \text{data}_1), \dots, (\text{op}_q, \text{addr}_q, \text{data}_q))$, handling each operation in the sequence consumes $g(N)$ computation steps.

It is immediate that worst-case overhead $g(N)$ directly implies amortized overhead of $g(N)$ but the converse is not necessarily true.

3.2 Basic Building Blocks

We briefly describe few functionalities that we use in our construction, and refer to [2] for their implementation:

1. **Intersperse**(X, Y) is an algorithm that takes two arrays, each is randomly shuffled, returns a randomly shuffled array, and runs in linear time (i.e., $O(|X| + |Y|)$). It obviously implements the ideal functionality $\mathcal{F}_{\text{Shuffle}}$ —which takes an array and randomly shuffled it.
2. **IntersperseRD**(X) takes an array that contains real and dummy elements, and is assumed that all real elements are shuffled among themselves, but there is no guarantee about the locations of the dummies in the array (e.g., they can all reside at the end of the array). The algorithm returns a randomly shuffled array, runs in linear time, and obviously implements the ideal functionality $\mathcal{F}_{\text{Shuffle}}$.
3. **Compaction:** Given an array in which some of the elements are distinguished, it moves all distinguished elements to the beginning of the array and runs in linear time.

3.3 Perfectly Oblivious 2-Key Dictionary

In this section we use known results to get a new oblivious dictionary-like data structure. Our structure, termed two-key dictionary, needs to support three operations: `Insert`, `PopKey`, and `PopTime`. Each element has a key k and a label t for “time”. The `Insert` operation takes the key k along with timestamp t and a value v and adds (k, t, v) to the dictionary, the k appears at most once in the dictionary (so that it overwrites if there is a previous tuple (k, t', v') for the same k). One can pop an element with key k by using `PopKey(k)`—the operation returns and removes the element with the key k . Analogously, `PopTime(t_1, t_2)` takes as input a time period $[t_1, t_2]$ and returns an element that is labeled with a timestamp t in the given period. This (reactive) functionality appears as [Functionality 3.3](#).

Functionality 3.3: $\mathcal{F}_{2\text{KeyDict}}$ - Dictionary Functionality

- Initialization of the state: let M be an empty list indexed by $k \in [K]$ for the given key space K , where all $M[k]$ are initialized as \perp .
 - $\mathcal{F}_{2\text{KeyDict}}.\text{Insert}(k, t, v)$:
 - **Input:** a key k , time $t \in \mathbb{N}$, and a value v , where k might be \perp , i.e., a dummy insertion.
 - **The procedure:**
 1. If $k \neq \perp$, set $M[k] := (t, v)$.
 - **Output:** The `Insert` operation has no output.
 - $\mathcal{F}_{2\text{KeyDict}}.\text{PopKey}(k)$:
 - **Input:** a key k (that might be \perp , i.e., dummy).
 - **The procedure:**
 1. Set $(t^*, v^*) := M[k]$ and then set $M[k] := \perp$.
 - **Output:** The value v^* .
 - $\mathcal{F}_{2\text{KeyDict}}.\text{PopTime}(t_1, t_2)$:
 - **Input:** time $t_1, t_2 \in \mathbb{N}$ such that $t_1 < t_2$.
 - **The procedure:**
 1. Let k be the smallest index such that $M[k] = (t^*, \cdot)$ for some $t^* \in [t_1, t_2]$. If no such k exists, set $v^* := \perp$. Otherwise, set $(t^*, v^*) := M[k]$ and then set $M[k] := \perp$.
 - **Output:** The value v^* .
-

Theorem 3.4. *Assume the tuple of (k, t, v) (i.e., key, time, and value) can be stored in a constant number of memory words. Assume further that the two-key dictionary needs to support at most n elements. There exists a perfectly oblivious implementation of functionality $\mathcal{F}_{2\text{KeyDict}}$ such that each operation `Insert`, `PopKey`, and `PopTime` takes $O(\log^4 n)$ time in the worst-case.*

Proof. The first step is to obtain a perfectly oblivious ORAM that has $O(\log^3 n)$ worst-case overhead when simulating a memory of size n . Such an ORAM can be obtained by applying the deamortization technique of Ostrovsky and Shoup [29]

on the (amortized) $O(\log^3 n)$ overhead perfect ORAM construction of Chan et al. [8]. It directly works (although never formally stated to the best of our knowledge) since the construction of Chan et al. is based exactly on the original hierarchical framework of Goldreich and Ostrovsky [16, 17] which was deamortized in Ostrovsky and Shoup [29].

Given this ORAM, it is straightforward to prove the theorem by just compiling a non-oblivious implementation of $\mathcal{F}_{2\text{KeyDict}}$. For the latter, we instantiate two balanced binary search trees (e.g., red-black tree), where the first tree orders elements according to the key k , and the second tree orders elements by the given time t . This implementation has logarithmic cost for each operation. Therefore, compiling it using the above worst-case perfect ORAM, we have a perfectly oblivious implementation of $\mathcal{F}_{2\text{KeyDict}}$ taking $O(\log^4 n)$ time in the worst case. \square

3.4 Oblivious Hash Table for Shuffled Inputs

Here we recall what an oblivious hash table is and the shuffled inputs assumption (taken from [2, Section 4.4]). An oblivious hash table is a (reactive) functionality that supports three operations: **Build**, **Lookup** and **Extract** that are defined as follows. The **Build** operation gets as input an array of items, **Lookup** is used to search for an item and then delete it, and finally **Extract** returns the “remaining” elements in the table. Obliviousness means, as usual, that the access patterns throughout the life time of the system should be unrelated to the elements in the array nor the values being searched for. We will achieve this guarantee *assuming the input to Build is random shuffled*.

Functionality 3.5: \mathcal{F}_{HT} - Hash Table Functionality for Non-Recurrent Lookups

$\mathcal{F}_{\text{HT}}.\text{Build}(\mathbf{I})$: The input

- **Input:** an input array $\mathbf{I} = (a_1, \dots, a_n)$ containing n elements, where each a_i is either dummy or a (key, value) pair denoted $(k_i, v_i) \in \{0, 1\}^D \times \{0, 1\}^D$ for some $D \in \mathbb{N}$. We assume that both the key and the value can be stored in $O(1)$ memory words, i.e., $D = O(w)$ where w denotes the word size.
- **The procedure:**
 1. Initialize the internal state **state** to (\mathbf{I}, \mathbf{P}) where $\mathbf{P} = \emptyset$. \mathbf{P} will store the keys that were already queried.
 2. **Output:** The Build operation has no input.

$\mathcal{F}_{\text{HT}}.\text{Lookup}(k)$:

- **Input:** a key $k \in \{0, 1\}^D \cup \{\perp\}$.
- **The procedure:**
 1. Parse the internal state as $\text{state} = (\mathbf{I}, \mathbf{P})$.
 2. If $k \in \mathbf{P}$ (i.e., k is a recurrent lookup) then halt and output **fail**. (Security is only guaranteed if no such recurrent lookup is performed, so in the construction we do not need to check this explicitly.)

3. If $k = \perp$ or $k \notin \mathbf{I}$ then set $v^* = \perp$.
 4. Otherwise, set $v^* = v$ where v is the value that corresponds to the key k in \mathbf{I} .
 5. Update $\mathbf{P} = \mathbf{P} \cup \{(k, v)\}$.
- **Output:** The element v^* .

$\mathcal{F}_{\text{HT}}.\text{Extract}()$:

- **The procedure:**
1. Parse the internal state $\text{state} = (\mathbf{I}, \mathbf{P})$.
 2. Define an array $\mathbf{I}' = (a'_1, \dots, a'_n)$ as follows. For $i \in [n]$ set $a'_i = a_i$ if $a_i = (k, v) \notin \mathbf{P}$. Otherwise, set $a'_i = \text{dummy}$.
 3. Shuffle \mathbf{I}' uniformly at random.
- **Output:** The array \mathbf{I}' .
-

The work of Asharov et al. [2, Corollary 8.9] shows a construction of a hash table, denoted as **CombHT**, with the following properties:

Theorem 3.6. *Assume that one-way functions exist. Then, for any $c \in \mathbb{N}$, there exists a construction, denoted as **CombHT**, that (computationally) obliviously implements the \mathcal{F}_{HT} functionality (Functionality 3.5) with the following properties:*

1. The input array is $\log^{9+c} \lambda \leq n \leq 2^\lambda$;
2. The input assumption is that the input array is randomly shuffled;
3. **Build** and **Extract** each take $O(n)$ time. **Build** outputs a main table and a stash of size $O(\log \lambda)$;
4. **Lookup** takes $O(1)$ time in addition to linearly scanning a stash of size $O(\log \lambda)$.

The high-level idea of this construction is given in the full version for completeness, and we refer to [2, Section 8.4] for full details.

4 Oblivious Deduplication in Linear Time

Consider two arrays X_1 of size n and X_2 of size $2n$, where it is guaranteed that at least half of the elements in X_2 are dummies,⁴ and the keys in each array are unique. In what follows we describe an algorithm for merging the (real) contents of the arrays while removing duplicates, preferring the ones in X_1 . That is, viewing the input arrays as sets, we compute $X_1 \cup X_2$ while preferring duplicate elements from X_1 . We start with the abstract functionality $\mathcal{F}_{\text{Dedup}}$ and then give our implementation.

⁴ One could easily modify our algorithm to work more generally for a list X_2 of size m which has at least n dummies and result with an array of size m . We chose to be concrete for simplicity.

The functionality $\mathcal{F}_{\text{Dedup}}$. The exact functionality is described next. It can be viewed as a non-efficient non-oblivious implementation. The input consists of an array X_1 of size n and an array X_2 of size $2n$. The array X_2 contains at most n real elements. Every key appears at most once in each input list (a key may appear once in each of the arrays with different associate values). The functionality does the following:

1. Initialize an array Y of size $2n$.
2. Copy to Y all real elements in both arrays X_1 and X_2 . If the two arrays contain the same key (with possibly different associated value), then remove the copy from X_2 and prefer the one in X_1 . Pad Y with dummies to be of size $2n$.
3. Uniformly shuffle Y and return it.

The main theorem of this section is stated next.

Theorem 4.1. *There is an algorithm that implements the functionality $\mathcal{F}_{\text{Dedup}}$ in time $O(n)$ for $n \geq \log^{11} \lambda$ (and with negligible error probability). The algorithm is computationally oblivious if one-way functions exist and if the input arrays are independently randomly shuffled.*

Recall the $O(n)$ time non-oblivious algorithm from Sect. 2—it is clearly non-oblivious and “naively” replacing the Cuckoo hash with an oblivious version thereof does not meet our goal since known constructions require $\omega(n)$ time for building. However, we do not necessarily need the full power of Cuckoo hashing since we are guaranteed that the input lists are shuffled. Therefore, instead of using Cuckoo hash, we use the hash table **CombHT** from Sect. 3.4 which has linear build time and otherwise behaves conceptually in a similar manner to “standard” oblivious Cuckoo hash: lookup is performed by a scan of a stash and $O(1)$ accesses to a “main table”. The idea therefore is conceptually in the same spirit, but we rearrange and then compose the procedures of **CombHT** (in a non-black-box way) to guarantee obliviousness. Namely in Step 3, for each duplicate that reside in the first hash table, we mark the element by its counterpart in the second hash table; then in Step 5, we are able to emulate identically the lookup procedures on the second table (even we perform no access on its stash). We refer the reader to the construction of **CombHT** in the full version for a comprehensive construction.

The algorithm $\text{Dedup}(X_1, X_2)$ works as follows:

1. Perform $T_1 := \text{HT.Build}(X_1)$ and $T_2 := \text{HT.Build}(X_2)$.
2. Denote $T_1 = (\text{sk}_1, \text{OBins}_1, \text{CombS}_{1,T}, \text{CombS}_{1,S}, \text{OF}_{1,T}, \text{OF}_{1,S})$ and $T_2 = (\text{sk}_2, \text{OBins}_2, \text{CombS}_{2,T}, \text{CombS}_{2,S}, \text{OF}_{2,T}, \text{OF}_{2,S})$.
3. Initialize an empty array L . Linearly scan $\text{OF}_{2,S}$ and $\text{Comb}_{2,S}$, and for each element (k, v) , perform the following:
 - (a) Perform a real lookup $(k', v') := T_1.\text{Lookup}(k)$. If k is found in T_1 then:
 - i Mark the element (k', v') at T_1 as “CombS” if k comes from $\text{CombS}_{2,S}$, or “OF” if k comes from $\text{OF}_{2,S}$.

- ii Mark the element (k, v) as “**accessed**” in T_2 .
 - (b) Write (k', v') to the next slot in L (including the mark when k is found). In the end of this loop, obviously shuffle L .
4. Perform $S'_1 := T_1.\text{Extract}()$. Then, perform $S_1 := \text{Intersperse}(S'_1 \| L)$.
 5. Linearly scan S_1 , and for each element (k, v) , perform the following:
 - (a) If k is marked as “**OF**”, then perform a real lookup $(k', v') := T_2.\text{Lookup}(k)$ while not scanning the stashes $(\text{OF}_{2,S}, \text{CombS}_{2,S})$ and proceeding as if k is found in $\text{OF}_{2,S}$.
 - (b) If k is marked as “**CombS**”, then perform a real lookup $(k', v') := T_2.\text{Lookup}(k)$ while not scanning the stashes $(\text{OF}_{2,S}, \text{CombS}_{2,S})$ and proceeding as if k is found in $\text{CombS}_{2,S}$.
 - (c) If k is not marked, then perform a real lookup $(k', v') := T_2.\text{Lookup}(k)$ while not scanning the stashes $(\text{OF}_{2,S}, \text{CombS}_{2,S})$ and proceeding as if k is not found in the stashes.
 6. Perform $S_2 := T_2.\text{Extract}()$ (recall that “**accessed**” elements in T_2 are not extracted).
 7. Run $Z := \text{Intersperse}(S_1 \| S_2)$. Run tight compaction on Z to move all dummy elements to the end. Truncate the array to be of size $2n$. Run IntersperseRD to randomly shuffle Z .
 8. Output Z .

The full proof of Theorem 4.1 appears in the full version. Here, we briefly argue that the efficiency is as required. In Step 3, we scan the stashes of T_2 and then for each element perform a $O(\log \lambda)$ -time lookup, and then we shuffle L . Since the stashes are of size $O(\log \lambda)$, the running time of this step is $O(\log^2 \lambda) \leq O(n)$. Step 5 consist of a linear scan of a list and then an $O(1)$ lookup on each item. Steps 4 and 6 consume $O(n)$ time. Step 7 consumes $O(n)$ time, as well. Overall, the overhead is linear in n , as needed.

5 The ORAM Construction with Worst Case Complexity

In this section we present our deamortized construction.

The combined stash technique. As we saw in Sect. 3.4, our hash table supports lookup in $O(1)$ time in addition to a lookup in a stash of size $O(\log \lambda)$. To allow faster lookups, constructions use “the combined stash” technique (see [2, 7, 20]). According to this technique, all stashes of all levels are combined into one global stash. Then, utilizing the fact that we look for the same element in all levels (or dummy lookup once the element is found), we have to search for the element only once in the global stash (instead of searching for it in $O(\log N)$ different stashes), and then spend just $O(1)$ lookup time per level.

As we mentioned in the introduction, the fact that there is a shared memory to all levels introduces some complications in the final construction. We therefore present our deamortized construction in two steps:

1. In Sect. 5.1 we look at a somewhat idealized construction in which the main building block is a hash table that takes $O(1)$ time per lookup, while it takes linear time for **Extract** and **Build** (on shuffled inputs). That is, “there is no stash”. We emphasize that we do not know how to realize such an oblivious hash table. Nevertheless, we describe this construction for aiding understanding and capturing the main ideas behind our deamortized construction.
2. In the full version we proceed to our final construction, in which the hash table is implemented as in Sect. 3.4, that is, $O(1)$ lookup time in addition to a scan of a stash of size $O(\log \lambda)$. To achieve effectively $O(1)$ time per lookup, we have also to use the combining stash technique as we described above.

5.1 Assuming Hash Table with $O(1)$ Lookup Time

In this section, we assume the existence of a construction of a hash table, denoted as HT, that achieves the following:

Assumption 5.1. *Assume that for any $c \in \mathbb{N}$ there exists a construction, denoted as HT that obviously implements the \mathcal{F}_{HT} functionality (Functionality 3.5) with the following properties:*

1. The input array is $\log^{9+c} \lambda \leq n \leq 2^\lambda$;
2. The input assumption is that the input array is randomly shuffled;
3. **Build** and **Extract** each take $O(n)$ time.
4. **Lookup** takes $O(1)$ time.

This is equivalent to Theorem 3.6 where the construction has no stash and **Lookup** takes worst-case $O(1)$ time. We proceed to the construction. We first start with the underlying primitives and the memory organization, and proceed to the specification of the construction.

Structure: Let $\ell = \lceil 11 \log \log \lambda \rceil$ and $L = \lceil \log N \rceil$.

1. Each level $i \in \{\ell + 1, \dots, L\}$ consists of four instances of HT as in Assumption 5.1, each of capacity 2^i . We denote the levels as $(\mathbf{A}_{\ell+1}^{\text{HF}}, \dots, \mathbf{A}_L^{\text{HF}})$, $(\mathbf{A}_{\ell+1}^{\text{F}}, \dots, \mathbf{A}_L^{\text{F}})$, $(\mathbf{B}_{\ell+1}^{\text{HF}}, \dots, \mathbf{B}_L^{\text{HF}})$ and $(\mathbf{B}_{\ell+1}^{\text{F}}, \dots, \mathbf{B}_L^{\text{F}})$.
2. Two perfect dictionaries (see Sect. 3.3), denoted as $\mathbf{A}_\ell, \mathbf{B}_\ell$, each of capacity $2^{\ell+1} + O(\log N \cdot \log \lambda)$. Each dictionary holds elements of the form $(\text{addr}, \text{data})$ where $\text{addr} \in [N]$, $\text{data} \in \{0, 1\}^w$.
3. Pointers $(\mathbf{A}_\ell, \dots, \mathbf{A}_L)$, $(\mathbf{B}_\ell, \dots, \mathbf{B}_L)$ where each \mathbf{A}_i points to either $\{\mathbf{A}_i^{\text{HF}}, \mathbf{A}_i^{\text{F}}, \text{Null}\}$ and each \mathbf{B}_i to $\{\mathbf{B}_i^{\text{HF}}, \mathbf{B}_i^{\text{F}}, \text{Null}\}$, where **Null** is a null pointer.
4. A global counter **ctr**, initialized as 0.

Construction 5.2: Oblivious RAM Access(op, addr, data)

- **Input:** op \in {read, write}, addr \in $[N]$ and data \in $\{0, 1\}^w$.
- **Secret state:** As above.
- **Initialization:** ctr is initialized to 0 as above, and all other data structures are initialized as empty.
- **The algorithm:**

Lookup:

1. Initialize found = false, data* = \perp .
2. Perform fetched := A_ℓ .PopKey(addr).
3. If fetched $\neq \perp$: then B_ℓ .PopKey(\perp).
Otherwise, fetched := B_ℓ .PopKey(addr).
4. If fetched $\neq \perp$: set found = true.
5. For each $i \in \{\ell + 1, \dots, L\}$ in increasing order, do (if A_i (or B_i resp.) is “null”, then let the result of Lookup be \perp):
 - (a) If found = false:
 - i. Set fetched := A_i .Lookup(addr).
 - ii. If fetched $\neq \perp$ then set found := true and data* := fetched.
 - (b) Else, perform A_i .Lookup(\perp).
 - (c) If found = false:
 - i. Set fetched := B_i .Lookup(addr).
 - ii. If fetched $\neq \perp$ then set found := true and data* := fetched.
 - (d) Else, perform B_i .Lookup(\perp).

Write back:

6. If found = false, i.e., this is the first time addr is being accessed, set data* = 0.
7. Let $(k, v) := (\text{addr}, \text{data}^*)$ if this is a read operation; else let $(k, v) := (\text{addr}, \text{data})$.
8. Insert (k, v) into A_ℓ and B_ℓ using Insert($k, \text{ctr} \bmod 2^{\ell+1}, v$).

Rebuild:

9. Increment ctr by 1.
10. For $i \in \{\ell + 1, \dots, L\}$:
 - (a) If $\text{ctr} \equiv 2^{i-2} \bmod 2^i$ then continue to 1-out-of-4 case:

	If ctr \equiv	0 mod 2^i	2^{i-2} mod 2^i	$2 \cdot 2^{i-2}$ mod 2^i	$3 \cdot 2^{i-2}$ mod 2^i
Set $A_i :=$		Null	A_i^{HF}	Null	A_i^{F}
Set $B_i :=$		B_i^{F}	B_i^{HF}	B_i^{HF}	Null
Start		RebuildHF(A_i^{HF})	RebuildHF(B_i^{HF})	RebuildF(A_i^{F})	RebuildF(B_i^{F})

By starting a task we mean to add the relevant task into the list Tasks. The procedures RebuildHF and RebuildF are defined below.

11. In a round robin fashion, for each task $t \in \text{Tasks}$, execute t.eachEpoch steps.
12. Return v .

Before proceeding, we refer the reader to depictions of the rebuilding scheduling in Figs. 2 and 3. In Step 10a, the schedule of the rebuild tasks is asymmetric (A_i^{HF} and A_i^{F} always start earlier than the B_i counterparts). This leads to the asymmetry between the setting of pointers A_i and B_i in Step 10a. Due to the asymmetry in schedule, there is a period such that both pointers A_i and B_i are available and storing distinct sets of elements (i.e., from $\text{ctr} \equiv 2^{i-2} \pmod{2^i}$ to $\text{ctr} \equiv 2 \cdot 2^{i-2} \pmod{2^i}$). Hence, Step 5 has to fetch `addr` in both A_i and B_i as their contents are distinct (our schedule is deterministic, but fetching `addr` in both tables is necessary as we do not know which table stores `addr`).

The procedure RebuildF. In this procedure, we build the table A_i^{F} from the two tables A_{i-1}^{F} and A_i^{HF} (similarly, B_i^{F} from B_{i-1}^{F} and B_i^{HF}). This is done by extracting the two tables, running `Dedup` (see Sect. 4) on the two tables, and then building the hash table. All those operations take linear work, and therefore we can spend $O(1)$ time per `Access` to the ORAM and finish the task in linear time. This is formalized in the `eachEpoch` variable.

In case the level to be rebuilt is $\ell + 1$, we extract all elements from the dictionary of level ℓ . This takes $O(\text{poly log log } N)$ per element. This will also be the `eachEpoch` value. That is, we spend $O(\text{poly log log } N)$ work for the rebuilding of level $\ell + 1$ with each `Access` to the ORAM.

RebuildF(C_i^{F}):

- **Input:** The task has input $C_i^{\text{F}} \in \{A_i^{\text{F}}, B_i^{\text{F}}\}$ for some index $i \in \{\ell + 1, \dots, L\}$.
- **eachEpoch:** The total time allocated to this task is 2^{i-2} .
 1. If $i = \ell + 1$: Let $W \in O(2^i \cdot \text{poly}(\text{log log } N))$ bound the total work of this procedure. Set `eachEpoch` = $W/2^{i-2} = \text{poly log log } N$.⁵
 2. If $i > \ell + 1$: The total work is $W \in O(2^i)$. Set `eachEpoch` = $W/2^{i-2} \in O(1)$.
- **The task:**
 1. If $i = \ell + 1$, run $C_{i-1}.\text{PopTime}(0, 2^\ell - 1)$ repeatedly for 2^ℓ times. That is, we extract all elements with $\text{ctr mod } 2^{\ell+1} \in [0, 2^\ell - 1]$, i.e., all elements that were added to the dictionary while building A_i^{HF} and B_i^{HF} . Let X be the list of popped elements, and then obviously shuffle X . Run $Y := C_i^{\text{HF}}.\text{Extract}()$.
 2. Else $i > \ell + 1$, run $X := C_{i-1}^{\text{F}}.\text{Extract}()$ and $Y := C_i^{\text{HF}}.\text{Extract}()$.
 3. Run $Z := \text{Dedup}(X, Y)$.
 4. Run $C_i^{\text{F}} := \text{HT.Build}(Z)$.

The procedure RebuildHF. In this procedure, we rebuild table A_i^{HF} from the contents of the table A_{i-1}^{F} (or B_i^{HF} from B_{i-1}^{F}). This is performed by adding dummy elements and building the next level. For $i > \ell + 1$ this requires linear work, and therefore we can spend $O(1)$ time per `Access` to the ORAM and finish the task in linear time. Likewise the case of `RebuildF`, the level $\ell + 1$ requires

⁵ Note that this implies that we run $\text{poly log log } N$ work per each access for the first level.

some more work but we have to finish also in linear time, so we spend more work with each access to the ORAM.

For the case of $i = L$ we do not simply build A_L^{HF} from A_{L-1}^{F} . Instead, we also have to merge the contents on A_L^{F} and A_{L-1}^{F} into A_L^{HF} . This is performed similarly to **RebuildF**: We first extract the two levels, run deduplication, and build level L .

RebuildHF(C_i^{HF}):

- **Input:** The task gets as input a table $C_i^{\text{HF}} \in \{A_i^{\text{HF}}, B_i^{\text{HF}}\}$ for some index $i \in \{\ell + 1, \dots, L\}$.
- **eachEpoch:** The total time allocated to this task is 2^{i-2} .
 1. If $i = \ell + 1$: Let $W \in O(2^i \cdot \text{poly}(\log \log N))$ bound the total work of this procedure. Set $\text{eachEpoch} = W/2^{i-2} = \text{poly} \log \log N$.
 2. If $i > \ell + 1$: The total work is $W \in O(2^i)$. Set $\text{eachEpoch} = W/2^{i-2} \in O(1)$.
- **The task:**
 1. If $i = L$:
 - (a) Run $X := C_{L-1}^{\text{F}}.\text{Extract}()$ and $Y := C_L^{\text{F}}.\text{Extract}()$.
 - (b) Run $Z := \text{Dedup}(X, Y)$.
 - (c) Run $C_L^{\text{HF}} := \text{HT}.\text{Build}(Z)$.
 2. Otherwise:
 - (a) If $i = \ell + 1$, run $C_{i-1}.\text{PopTime}(2^\ell, 2^{\ell+1} - 1)$ repeatedly for 2^ℓ times. That is, we extract all elements with $\text{ctr} \bmod 2^{\ell+1} \in [2^\ell, 2^{\ell+1} - 1]$, i.e., all elements that were added to the dictionary while building A_i^{F} and B_i^{F} . Let X be the list of popped elements, and then obliviously shuffle X .
 - (b) Else $i > \ell + 1$, run $X := C_{i-1}^{\text{F}}.\text{Extract}()$.
 - (c) Initialize an array Y of 2^{i-1} dummies.
 - (d) Intersperse X and Y into Z and run $C_i^{\text{HF}}.\text{Build}(Z)$.

Analysis. We next prove the following theorem:

Theorem 5.3. *Let N be the capacity of the ORAM and let $\lambda \in \mathbb{N}$ be a security parameter. Assuming the existence of HT as in Assumption 5.1, Construction 5.2 obliviously implements the ORAM functionality, and each Access takes $O(\log N + \log^4 \log \lambda)$ in the worst case.*

Proof. We start with the efficiency analysis. Each access requires two lookups (**PopKey**) at the dictionaries A_ℓ, B_ℓ (Steps 2 and 3) and writing back to the two dictionaries (Step 8). Each dictionary contains at most $2^\ell \leq \log^{12} \lambda$, and each access costs $O(\log^4 \log \lambda)$ time (see Sect. 3.3).

Then, we perform one access to each one of the tables $A_{\ell+1}, \dots, A_L, B_{\ell+1}, \dots, B_L$, each takes $O(1)$ time by Assumption 5.1, and overall it takes $O(\log N)$ times. So overall, the lookup and write back take $O(\log N + \log^4 \log \lambda)$ work.

In the rebuild process, by construction we have exactly one task being rebuilt in each level, and start the next task only when the previous one finishes. It is easy to see that each process takes a linear time in the size of the level, and

therefore we spend $O(1)$ per task with each **Access** to the ORAM, except for level $\ell + 1$. The procedures for level $\ell + 1$ require 2^ℓ accesses to the dictionaries, each translates to $O(\log^4 \log \lambda)$ work (total $O(2^\ell \cdot \log^4 \log \lambda)$), and oblivious shuffle of a list of size 2^ℓ which can also be implemented in total $O(2^\ell \cdot \log^4 \log \lambda)$. Therefore, we spend $O(\log N + \log^4 \log \lambda)$ work for the rebuilding of all levels, combined.

Security. Since the ORAM functionality is deterministic, it is enough to separately consider correctness and obliviousness. We show here obliviousness, and then discuss correctness.

We show security in the hybrid model where we invoke $\mathcal{F}_{2\text{KeyDict}}$, \mathcal{F}_{HT} , $\mathcal{F}_{\text{Dedup}}$, $\mathcal{F}_{\text{Shuffle}}$ instead of oblivious dictionary, oblivious hash table, oblivious deduplication and intersperse, respectively. Replacing all ideal functionalities with the corresponding construction is straightforward using the composition theorem.

It is easy to simulate Construction 5.2: We access the two dictionaries, and then access the two hash tables in each level and finally write back to the dictionaries. The rebuild process and which hash table we use has a public schedule known to the adversary. Likewise which tasks are currently running.

We now show how to simulate the two procedures: **RebuildF** and **RebuildHF**. For the case of $i > \ell + 1$, in **RebuildF**: We just have two ideal calls to **Extract**. Since **Extract** returns an oblivious permutation of the element in the hash table, this implies that the input assumption of **Dedup** is preserved. We then obtain an array of size 2^i which is randomly shuffled and therefore the input assumption of \mathcal{F}_{HT} is preserved. Simulation is just these three ideal calls. Simulating **RebuildHF** is similar for the case of $i = L$, and for the case of $i \in \{\ell + 2, \dots, L - 1\}$ it is also just ideal calls to $\mathcal{F}_{\text{HT}}.\text{Extract}$, $\mathcal{F}_{\text{Shuffle}}$ (to intersperse the two arrays) and $\mathcal{F}_{\text{HT}}.\text{Build}$. As for $i = \ell + 1$, in both procedures we have ideal calls to $\mathcal{F}_{2\text{KeyDict}}$ and we shuffle the output so that input assumptions are preserved.

Correctness. We also prove the correctness in the hybrid model, and our goal is to show that every **Access** to an address **addr** reads the data that was most recently written to **addr**, i.e., satisfying the ORAM functionalities: Each **Access**(**read**, **addr**, \perp) for a given **addr** will have the answer **data**, according to the last operation **Access**(**write**, **addr**, **data**) that was given to the ORAM (with the same **addr**). We begin with describing two invariants in Definitions 5.4 and 5.5 and show that they hold.

Definition 5.4 (Vertical invariant). *Fixing any $\text{addr} \in [N]$, we say that $(\text{addr}, \text{data})$ is the freshest version at some given time ctr if the pair $(\text{addr}, \text{data})$ is the most recent pair having **addr** read or written by **Access** operation to the ORAM. Then, for every $\text{addr} \in [N]$, it holds that*

- every level in the hierarchy $H_A := \{A_i\}_{i \in [\ell, L]}$ consists of at most one version of **addr**, and
- Among all levels in $\{A_i^{\text{HF}}, A_i^{\text{F}}\}_{i \in [\ell, L]}$ that contain **addr** (in which some might be rebuilt and unavailable), the freshest version of **addr** must reside in the smallest level.

This holds symmetrically for hierarchy $H_B := \{B_i\}_{i \in [\ell, L]}$.

Definition 5.5 (Horizontal invariant). *For every $\text{addr} \in [N]$, it holds that the freshest version of addr must fall in one of the following cases:*

1. *It is in the same level of the two hierarchies, i.e., it is in both A_i and B_i for some $i \in [\ell, L]$.*
2. *It is in $\text{RebuildF}(A_i^F)$ and B_i^{HF} for some $i \in [\ell + 1, L]$, and $B_i = B_i^{\text{HF}}$.*
3. *It is in A_i^F and $\text{RebuildF}(B_i^F)$ for some $i \in [\ell + 1, L]$, and $A_i = A_i^F$.*
4. *It is in B_i and either in $\text{RebuildHF}(A_{i+1}^{\text{HF}})$ or in $\text{RebuildF}(A_{i+1}^F)$ for some $i \in [\ell, L - 1]$.*
5. *It is in A_{i+1} and either in $\text{RebuildHF}(B_{i+1}^{\text{HF}})$ or in $\text{RebuildF}(B_{i+1}^F)$ for some $i \in [\ell, L - 1]$.*

The invariants imply correctness. Using the above vertical and horizontal invariants (whose proofs are below), it suffices to syntactically check the correctness: we list all possible locations of the newest version below and conclude the correctness of *Access*.

- In both A_ℓ and B_ℓ : the element from A_ℓ is outputted (following Step 2).
- In B_ℓ while $A_{\ell+1}$ is rebuilding ($A_{\ell+1}$ takes elements from A_ℓ but $A_{\ell+1}$ is still unavailable): the element from B_ℓ is outputted (following Step 3).
- In $A_{\ell+1}$ while $B_{\ell+1}$ is rebuilding ($B_{\ell+1}$ takes elements from B_ℓ but $B_{\ell+1}$ is still unavailable): the element from $A_{\ell+1}$ is outputted (Step 5(a)i).
- In both A_i and B_i , $i \in [\ell + 1, L]$: the element from A_i is outputted (Step 5(a)i).
- In B_i while A_{i+1} is rebuilding down, for $i \in [\ell + 1, L - 1]$: there are two cases, either A_{i-1} has finished its rebuild down to A_i , or A_{i-1} has not yet. In both cases, the element from B_i is outputted by Step 5(c)i.
- In A_{i+1} while B_i is rebuilding down, $i \in [\ell + 1, L - 1]$: the element from A_{i+1} is outputted (Step 5(a)i).

Notice that for two addresses $\text{addr}, \text{addr}'$, it may happen that addr is in Case 4 for some i and that addr' is in Case 5 for $i' = i - 1$. This means addr is in B_i while addr' is in A_i , so that both A_i and B_i are available for lookup, but they have disjoint contents addr and addr' . This special case explains the reason we perform lookup on both A_i and B_i in Steps 5(a)i and 5(c)i.

In the full version, we prove two lemmas showing that both the vertical (Definition 5.4) and the horizontal invariant (Definition 5.5) hold in the construction. This concludes the proof. □

Moreover, in the full version we also use the combined stash technique and show how to deamortize it as well. We show:

Theorem 5.6. *Let N be the capacity of the ORAM and let $\lambda \in \mathbb{N}$ be a security parameter. Assuming the existence of one-way functions, the construction described above obliviously implements the ORAM functionality. The construction has $O(\log N + \log^4 \log \lambda)$ worst case overhead.*

Acknowledgments. This work is supported in part by a DARPA Brandeis award, by NSF under the award numbers CNS-1601879, CNS-2044679, by Packard Fellowship, an ONR YIP award, by the ISRAEL SCIENCE FOUNDATION (grants No. 2439/20 and 1774/20), by an Alon Young Faculty Fellowship, and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister’s Office. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.

A Figures

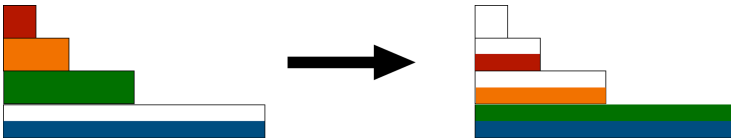


Fig. 1. The rebuild process of [3]: The first three levels are “full” and the forth is the first level which is “half full”. Each level is pushed down, while levels 3 and 4 are merged. After this operation, the first level is empty, two levels are “half full” and the last level is full.

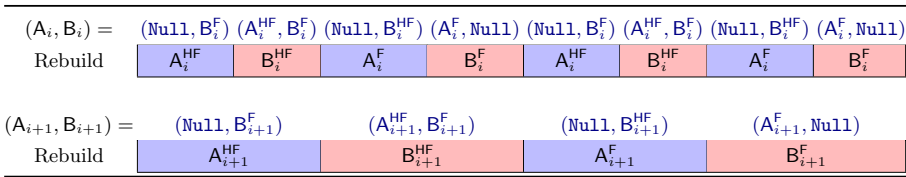


Fig. 2. The Rebuild process (for levels i and $i + 1$), demonstrating which table is being rebuilt at each stage and which tables we lookup in with each access. The timeline goes left-to-right, each colored box is rebuilding the enclosed table, and the left/right side of the box denotes the starting/ending time of the rebuild. Notice that the rebuild at level $i + 1$ changes the status in both levels i and $i + 1$, e.g., the starting of B_{i+1}^F (on the bottom-right) switches both B_i and B_{i+1} to Null, and its ending assigns $B_{i+1} := B_{i+1}^F$.

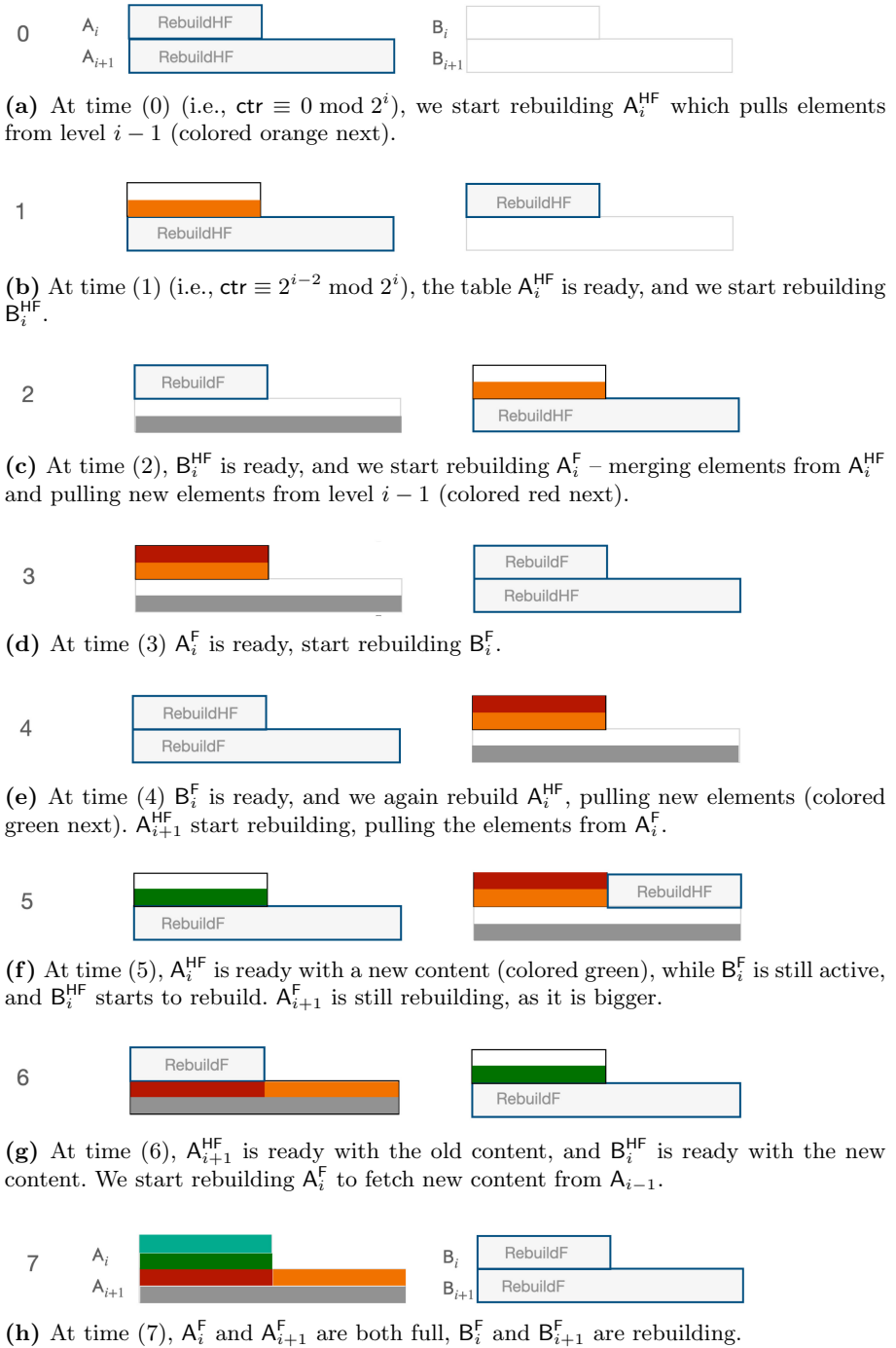


Fig. 3. The rebuilding process. A_i^{HF} and A_i^{F} are both shown in the same table, likewise B_i^{HF} and B_i^{F} . (Color figure online)

References

1. Ajtai, M., Komlós, J., Szemerédi, E.: An $O(n \log n)$ sorting network. In: ACM STOC, pp. 1–9 (1983)
2. Asharov, G., Komargodski, I., Lin, W.-K., Nayak, K., Peserico, E., Shi, E.: OptORAMA: optimal oblivious RAM. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 403–432. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_14
3. Asharov, G., Komargodski, I., Lin, W., Peserico, E., Shi, E.: Optimal oblivious parallel RAM. IACR ePrint Archive **2020**, 1292 (2020)
4. Bindschaedler, V., Naveed, M., Pan, X., Wang, X., Huang, Y.: Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward. In: ACM CCS, pp. 837–849 (2015)
5. Boyle, E., Naor, M.: Is there an oblivious RAM lower bound? In: ITCS, pp. 357–368 (2016)
6. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: CCS, pp. 668–679 (2015)
7. Chan, T.-H.H., Guo, Y., Lin, W.-K., Shi, E.: Oblivious hashing revisited, and applications to asymptotically efficient ORAM and OPRAM. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 660–690. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_23
8. Chan, T.-H.H., Nayak, K., Shi, E.: Perfectly secure oblivious parallel RAM. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 636–668. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_23
9. Hubert Chan, T.-H., Shi, E.: Circuit OPRAM: unifying statistically and computationally secure ORAMs and OPRAMs. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 72–107. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_3
10. Chung, K.-M., Liu, Z., Pass, R.: Statistically-secure ORAM with $\tilde{O}(\log^2 n)$ overhead. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 62–81. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_4
11. Dittmer, S., Ostrovsky, R.: Oblivious tight compaction in $o(n)$ time with smaller constant. In: SCN, pp. 253–274 (2020)
12. Fletcher, C.W., Dijk, M.V., Devadas, S.: A secure processor architecture for encrypted computation on untrusted programs. In: ACM Workshop on Scalable Trusted Computing, pp. 3–8 (2012)
13. Fletcher, C.W., Ren, L., Kwon, A., van Dijk, M., Devadas, S.: Freecursive ORAM: [nearly] free recursion and integrity verification for position-based oblivious RAM. In: ASPLOS, pp. 103–116 (2015)
14. Fredman, M.L., Willard, D.E.: Surpassing the information theoretic bound with fusion trees. *J. Comput. Syst. Sci.* **47**(3), 424–436 (1993)
15. Gentry, C., Halevi, S., Judla, C., Raykova, M.: Private database access with HE-over-ORAM architecture. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 172–191. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_9
16. Goldreich, O.: Towards a theory of software protection and simulation by oblivious rams. In: ACM STOC, pp. 182–194 (1987)
17. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. *J. ACM* **43**(3), 431–473 (1996)

18. Goodrich, M.T., Mitzenmacher, M.: Privacy-preserving access of outsourced data via oblivious RAM simulation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 576–587. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_46
19. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Oblivious ram simulation with efficient worst-case access overhead. In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW 2011, pp. 95–100 (2011)
20. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Privacy-preserving group data access via stateless oblivious RAM simulation. In: SODA, pp. 157–167 (2012)
21. Grubbs, P., McPherson, R., Naveed, M., Ristenpart, T., Shmatikov, V.: Breaking web applications built on top of encrypted data. In: CCS, pp. 1353–1364 (2016)
22. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: 19th Annual Network and Distributed System Security Symposium, NDSS (2012)
23. Komargodski, I., Lin, W.: Lower bound for oblivious RAM with large cells. IACR Cryptology ePrint Archive **2020**, 1132 (2020)
24. Kushilevitz, E., Lu, S., Ostrovsky, R.: On the (in)security of hash-based oblivious RAM and a new balancing scheme. In: SODA, pp. 143–156 (2012)
25. Larsen, K.G., Nielsen, J.B.: Yes, there is an oblivious RAM lower bound! In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 523–542. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_18
26. Liu, C., Wang, X.S., Nayak, K., Huang, Y., Shi, E.: OblivM: a programming framework for secure computation. In: IEEE S&P (2015)
27. Lu, S., Ostrovsky, R.: Distributed oblivious RAM for secure two-party computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 377–396. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_22
28. Maas, M., et al.: PHANTOM: practical oblivious computation in a secure processor. In: ACM CCS, pp. 311–324 (2013)
29. Ostrovsky, R., Shoup, V.: Private information storage. In: ACM STOC, pp. 294–303 (1997)
30. Patel, S., Persiano, G., Raykova, M., Yeo, K.: Panorama: oblivious RAM with logarithmic overhead. In: IEEE FOCS (2018)
31. Ren, L., Yu, X., Fletcher, C.W., van Dijk, M., Devadas, S.: Design space exploration and optimization of path oblivious RAM in secure processors. In: The 40th Annual International Symposium on Computer Architecture, ISCA, pp. 571–582 (2013)
32. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_11
33. Stefanov, E., et al.: Path ORAM: an extremely simple oblivious RAM protocol. In: ACM CCS, pp. 299–310 (2013)
34. Stefanov, E., Shi, E.: ObliviStore: high performance oblivious cloud storage. In: IEEE S&P, pp. 253–267 (2013)
35. Stefanov, E., Shi, E., Song, D.X.: Towards practical oblivious RAM. In: NDSS (2012)
36. Thorup, M.: Randomized sorting in $O(n \log \log n)$ time and linear space using addition, shift, and bit-wise Boolean operations. J. Algorithms **42**(2), 205–230 (2002)

37. Wang, X., Chan, T.H., Shi, E.: Circuit ORAM: on tightness of the Goldreich-Ostrovsky lower bound. In: ACM CCS, pp. 850–861 (2015)
38. Wang, X.S., Huang, Y., Chan, T.H., Shelat, A., Shi, E.: SCORAM: oblivious RAM for secure computation. In: ACM CCS, pp. 191–202 (2014)
39. Williams, P., Sion, R., Tomescu, A.: PrivateFS: a parallel oblivious file system. In: ACM CCS (2012)
40. Zahur, S., et al.: Revisiting square-root ORAM: efficient random access in multi-party computation. In: IEEE S&P, pp. 218–234 (2016)
41. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: The power of file-injection attacks on searchable encryption. In: USENIX, pp. 707–720 (2016)



Puncturable Pseudorandom Sets and Private Information Retrieval with Near-Optimal Online Bandwidth and Time

Elaine Shi¹(✉), Waqar Aqeel², Balakrishnan Chandrasekaran³,
and Bruce Maggs²

¹ CMU, Pittsburgh, USA

`runting@cs.cmu.edu`

² Duke, Durham, USA

`{waqeel, bmm}@cs.duke.edu`

³ Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
`b.chandrasekaran@vu.nl`

Abstract. Imagine one or more non-colluding servers each holding a large public database, e.g., the repository of DNS entries. Clients would like to access entries in this database without disclosing their queries to the servers. Classical private information retrieval (PIR) schemes achieve polylogarithmic bandwidth per query, but require the server to perform linear computation per query, which is a significant barrier towards deployment.

Several recent works showed, however, that by introducing a one-time, per-client, off-line preprocessing phase, an *unbounded* number of client queries can be subsequently served with sublinear online computation time per query (and the cost of the preprocessing can be amortized over the unboundedly many queries). Existing preprocessing PIR schemes (supporting unbounded queries), unfortunately, make undesirable trade-offs to achieve sublinear online computation: they are either significantly non-optimal in online time or bandwidth, or require the servers to store a linear amount of state per client or even per query, or require polylogarithmically many non-colluding servers.

We propose a novel 2-server preprocessing PIR scheme that achieves $\tilde{O}(\sqrt{n})$ online computation per query and $\tilde{O}(\sqrt{n})$ client storage, while preserving the polylogarithmic online bandwidth of classical PIR schemes. Both the online bandwidth and computation are optimal up to a polylogarithmic factor. In our construction, each server stores only the original database and nothing extra, and each online query is served within a single round trip. Our construction relies on the standard LWE assumption. As an important stepping stone, we propose new, more generalized definitions for a cryptographic object called a Privately Puncturable Pseudorandom Set, and give novel constructions that depart significantly from prior approaches.

Please read the online full version [47] for complete details and proofs.

© International Association for Cryptologic Research 2021

T. Malkin and C. Peikert (Eds.): CRYPTO 2021, LNCS 12828, pp. 641–669, 2021.

https://doi.org/10.1007/978-3-030-84259-8_22

1 Introduction

Imagine that a service provider has a large public database, DB, and is serving clients who request records from DB. For example, in a search-engine scenario each entry in DB may be the search result for a specific keyword; in the DNS scenario, each entry contains the records for a specific domain name. Without loss of generality, we may assume that the database $DB \in \{0, 1\}^n$ is an array of bits indexed by $\{0, 1, \dots, n-1\}$, and a client’s query is an index $i \in \{0, 1, \dots, n-1\}$ into DB¹. Although the database itself is public, the clients wish to hide their queries from the server. This problem has been studied in a beautiful line of work called Private Information Retrieval (PIR), first formulated by Chor, Goldreich, Kushilevitz, and Sudan [18, 19]. Since then, a rich line of work [4, 9, 10, 13, 16, 17, 21, 23, 24, 26, 28, 32, 34, 35, 37, 38, 40, 42–44] has improved the original construction of Chor et al. [18]. This paper focuses on *2-server PIR*, i.e., there are two non-colluding servers, and the goal is to prevent each individual server from learning anything about the clients’ actual queries.

Single- or multi-server PIR schemes with *polylogarithmic* bandwidth (bits sent per query) and *linear server work* per query are well known [9, 10, 13, 16, 17, 24, 28, 32, 34, 37, 38, 42–44]. While these PIR schemes are elegant in construction and achieve non-trivial asymptotic bounds, the prohibitive server running time per query is a significant barrier towards practical deployment. For example, in our motivating applications, the database may have billions or trillions of entries. Unfortunately, in the original formulation phrased by Chor et al. [18], linear server work is required to achieve privacy [6]—intuitively, if there is a location that the server does not need to read, the query is definitely not looking for that location. To avoid this drawback, a promising direction has been suggested by a few recent works [6, 20], namely, PIR with *preprocessing*. In PIR with preprocessing, clients and servers are allowed to perform one-time offline preprocessing. After preprocessing, the PIR scheme should support an *unbounded* number of queries from each client. The cost of the offline preprocessing can thus be amortized “away” over sufficiently many queries, and we can hope for *sublinear amortized (i.e., online) running time* per query.

Preprocessing PIR was considered in several prior works [6, 38, 44]. Beimel, Ishai, and Malkin [6] were the first to suggest using preprocessing to reduce the server’s online computation. They constructed a statistically secure 2-server PIR scheme with n^ϵ online bandwidth and running time for some constant $\epsilon \in (0, 1)$ by having the servers preprocess the n -bit DB into an encoded version of $\text{poly}(n)$ bits. The line of work on preprocessing PIRs culminated in the elegant work by Corrigan-Gibbs and Kogan [20], who showed that, assuming one-way functions, there is a 2-server preprocessing PIR scheme with $O(\sqrt{n})$ online bandwidth and running time (ignoring the dependence on the security parameter). In their scheme the servers store only the original database DB and nothing extra, but each client needs to store a “hint” of size $O(\sqrt{n})$. Corrigan-Gibbs and Kogan [20] also proved that the $O(\sqrt{n})$ online computation is optimal, assuming that the

¹ If the query is a keyword or domain name, it can be hashed to an index, and if each entry has multiple bits, we can treat it as retrieving multiple indices.

client downloads only $O(\sqrt{n})$ amount of information from the server during preprocessing and that the servers store only the unencoded database (and the proof works by reducing PIR to Yao’s Box problem [53]). The main drawback with their scheme is the significantly non-optimal $O(\sqrt{n})$ online bandwidth which is also much worse than classical PIR without preprocessing.

Given the state of affairs for preprocessing PIR, we ask the following question:

Can we construct a preprocessing PIR scheme that is simultaneously optimal in online bandwidth and online time?

Before we present our results and contributions, we point out a couple of important desiderata and clarify the problem statement:

- *Unbounded query setting.* First, we want the PIR scheme to support an *unbounded* number of queries after a one-time processing. This is necessary in the vast majority of conceivable applications (e.g., oblivious DNS [1, 49], oblivious Safe Browsing [2], the four excellent use cases in the Splinter work [52], and other applications [3, 4]). Unsurprisingly, state-of-the-art PIR implementations invariably support unbounded number of queries too [3, 4, 52]. Without the unbounded requirement, there is indeed a scheme with $O(\sqrt{n})$ online computation and $\tilde{O}(1)$ online bandwidth shown in the same work of Corrigan-Gibbs and Kogan [20]—unfortunately, this scheme supports only a single query after the preprocessing, and thus the linear preprocessing cost should be charged to each query, and cannot be amortized over multiple queries.
- *No per-client server state.* Second, the server should not have to store per-client state. There are alternative solutions if we let the server store per-client state (and often $O(n)$ state per client). For example, one strawman candidate is to use an Oblivious RAM (ORAM) scheme [29, 31, 48]. During the offline phase, the client downloads the database from the server and uses a secret key to compile the database into an ORAM which is then stored on the server. This would allow queries to be supported in polylogarithmic running time and bandwidth per query, and constant roundtrips (provided the server can perform computation) [22, 25, 27, 39]. Unfortunately, $\Omega(n)$ per-client state on the server would clearly be a barrier towards practicality in some motivating applications. Similarly, the recent doubly-efficient (1-server) PIR constructions in the designated-client setting [11, 15] also suffers from the same drawback, although they remove the need for clients to store persistent state. A doubly-efficient PIR construction in the public-client setting promises to remove the $O(n)$ per-client state at the server. Unfortunately, the only known such construction relies on virtual blackbox (VBB) obfuscation which is known to be impossible [5]. We compare with additional related works in Sect. 7.

Besides the above, we also want the client-side storage to be small—if the client could store the entire database, then there is no need to talk to the server.

Our results and contributions. We answer the above question affirmatively, assuming Learning With Errors (LWE) [45]. Our scheme employs two servers, a “left” server and a “right” server and, at a high level, works as follows.

- During the offline preprocessing phase, each client sends a single message of size roughly $\tilde{O}(\sqrt{n})$ to the left server². The left server responds with a *hint* of $\tilde{O}(\sqrt{n})$ bits, which is stored by the client. Then online queries begin.
- For each online query, the client sends a single poly-logarithmically sized message to each server in parallel. In particular, the message sent to the right server is used for answering the query. Using its locally stored hint and the right server’s response, the client can reconstruct the correct answer to the query except with negligible probability. The message sent to the left server is used to partially “refresh” the client’s hint. The client uses the answer from the left server and the outcome of the present query to update one entry in the $\tilde{O}(\sqrt{n})$ -sized hint it stores.

More formally, we prove the following theorem:

Theorem 1 (2-server preprocessing PIR). *Assuming the Learning With Errors (LWE) assumption, there exists a 2-server preprocessing PIR scheme that satisfies the following performance bounds:*

- the offline server running time is $\tilde{O}(n)$; the offline client running time and bandwidth is $\tilde{O}(\sqrt{n})$.
- the online server and client time per query is $\tilde{O}(\sqrt{n})$; the online bandwidth per query is $\tilde{O}(1)$.
- each online query can be accomplished in a single roundtrip, that is, the client sends a single message to each server in parallel, and reconstructs the answer from the two servers’ responses respectively; and
- each server needs to store only the original database DB and no extra information; each client needs to store $\tilde{O}(\sqrt{n})$ bits of information.

Due to the lower bound of Corrigan-Gibbs and Kogan [20], our scheme’s total online time is *optimal up to poly-logarithmic factors*, assuming that the client downloads only approximately \sqrt{n} amount of information from the server during preprocessing. In comparison, the prior state-of-the-art scheme [20] can achieve optimal online computation, but their \sqrt{n} online bandwidth is significantly non-optimal. We improve their bandwidth consumption by a roughly \sqrt{n} factor, and thus achieve near optimality in both online computation and bandwidth. Table 1 compares our result with the most relevant prior work.

Theorem 1 does not give the exact constant c in the hidden $\log^c n$ factor; however, in the online full version [47], we give a more careful analysis of the concrete constants. Specifically, we show that with some fine-tuning, we can get the following more precise asymptotical performance where $\alpha(\lambda)$ denotes an arbitrarily small super-constant function: the offline server time is $O(n \log^2 n \log \lambda) \cdot \alpha(\lambda)$, the offline client time is $O(\sqrt{n} \log^2 n \log \lambda) \cdot \alpha(\lambda)$, and the offline client bandwidth is $O(\sqrt{n} \log^2 n \log \lambda) \cdot \alpha(\lambda)$. Moreover, the online client time per query is

² The $\tilde{O}(\cdot)$ notation hides polylogarithmic factors and dependence on the security parameter.

Table 1. Comparison with prior schemes. Includes only schemes where the servers need not store per-client state, has sublinear online time, and supports an unbounded number of queries (possibly after a one-time preprocessing). Sections 1 and 7 review additional related work in the broader design space, when we are willing to relax these desiderata. “C-Time”, “S-Time”, and “BW” denote client time, server time, and bandwidth, respectively. “OLDC” means oblivious locally decodable codes, and “VBB Obf.” means virtual-blackbox obfuscation. $\epsilon \in (0, 1)$ is a constant.

★: Beimel et al. [6] requires the servers to store a large $\text{poly}(n)$ amount of state.

Scheme	#server	Assumpt.	Offline			Online		
			C-Time	S-Time	BW	C-Time	S-Time	BW
[6]★	2	None	0	$\text{poly}(n)$	0	n^ϵ	n^ϵ	n^ϵ
[20]	2	OWF	$O(\sqrt{n})$	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
	2	OWF	$O(\sqrt{n})$	$O(n)$	$O(\sqrt{n})$	$O(n^{5/6})$	$O(\sqrt{n})$	$\tilde{O}(1)$
[11]	1	OLDC, VBB Obf.	0	0	0	n^ϵ	n^ϵ	n^ϵ
Our PIR	2	LWE	$\tilde{O}(\sqrt{n})$	$\tilde{O}(n)$	$\tilde{O}(\sqrt{n})$	$\tilde{O}(\sqrt{n})$	$\tilde{O}(\sqrt{n})$	$\tilde{O}(1)$
LB [20]	-	-	-	-	n/β	-	β	-

$O(\sqrt{n} \log^2 n \log \lambda) \cdot \alpha(\lambda)$, the online server runtime is $O(\sqrt{n} \log n \log \lambda) \cdot \alpha(\lambda)$, and the online bandwidth per query is $O(\log n \cdot \log \lambda) \cdot \alpha(\lambda)$.

Furthermore, in the online full version [47], we also discuss how to tune the parameters to get near optimality of the online bandwidth and computation, for every choice of offline bandwidth, in light of the known lower bound [20].

Remark 1. Like in earlier works [20], for simplicity, in our asymptotical performance bounds, we hide a security parameter $\chi(\lambda)$ factor that is related to the strength of the LWE assumption. If we assume standard polynomial security, $\chi(\lambda)$ is polynomially bounded in λ ; if we assume subexponential security, $\chi(\lambda)$ is poly-logarithmic in λ .

Technical highlight. Our 2-server preprocessing PIR scheme is inspired by the very recent work of Corrigan-Gibbs and Kogan [20]. At a high level, their work shows how to construct a 2-server preprocessing PIR scheme using a cryptographic object which they call a Puncturable Pseudorandom Set (PRSet). A PRSet scheme provides an algorithm for generating a secret key sk that can be used to generate a pseudorandom subset $\text{Set}(\text{sk}) \subseteq \{0, 1, \dots, n - 1\}$; sk thus serves as a succinct representation of the set $\text{Set}(\text{sk})$. Further, there is an efficient puncturing algorithm: suppose some element $x \in \text{Set}(\text{sk})$, then $\text{Puncture}(\text{sk}, x)$ outputs a punctured key sk_x that effectively removes x from the set, i.e., $\text{Set}(\text{sk}_x) = \text{Set}(\text{sk}) \setminus \{x\}$.

Unfortunately the Corrigan-Gibbs and Kogan [20] PRSet scheme is not efficient in all dimensions, namely, set enumeration time, membership test time, and punctured key size. As a tradeoff, they opt for efficient set enumeration and efficient membership test, allowing their PIR scheme to achieve roughly

\sqrt{n} online running time. Their PRSet scheme, however, adopts a trivial puncturing algorithm. The punctured key is simply the entire punctured set itself minus the element x to be removed, which causes their online bandwidth to be roughly \sqrt{n} , which is asymptotically worse than classical PIR schemes without preprocessing [10, 13, 17, 24, 26, 28, 37, 38, 42].

To achieve our stated result, an important stepping stone is to construct a new Privately Puncturable Pseudorandom Set (PRSet) that is efficient in all dimensions. Unfortunately, as explained in Sect. 2, these requirements seem to be inherently conflicting, and we were not able to directly reconcile them—likely Corrigan-Gibbs and Kogan [20] encountered the same barriers.

Our key insight is to observe that the Corrigan-Gibbs and Kogan formulation of a PRSet scheme seems too restrictive. We generalize their PRSet abstraction in the following ways.

1. *Emulating a customized sampling distribution.* Corrigan-Gibbs and Kogan consider only PRSet schemes that emulate simple distributions, such as sampling a random \sqrt{n} -sized subset among n elements, or sampling each element at random with probability $1/\sqrt{n}$. By contrast, we generalize the PRSet definition to allow it to emulate an arbitrary distribution of choice. Later we discuss the challenges of choosing this distribution.
2. *Relaxed correctness definition.* Corrigan-Gibbs and Kogan’s definition insists on almost-always correctness. We observe that a weaker notion, which we call “occasional correctness,” is sufficient for obtaining a 2-server preprocessing PIR (since our PIR construction relies on parallel repetition to amplify the correctness to $1 - \text{negl}(\lambda)$) where λ denotes the *security parameter* globally. Specifically, we want the puncturing algorithm to remove the point x being punctured, and only the point x —but we only need this to happen with considerable but not overwhelming probability.

Therefore, one technical contribution we make is to devise a more generalized/relaxed abstraction of a Privately Puncturable Pseudorandom Set (PRSet) scheme that is suitable and sufficient for constructing an efficient 2-server preprocessing PIR. To do so, we need to identify an appropriate sampling distribution that the PRSet should emulate. In our carefully chosen distribution, each element from $\{0, 1, \dots, n-1\}$ is included in the set with roughly $1/(\sqrt{n} \cdot \text{poly log } n)$ probability, but the sampling is not completely independent among the elements. For example, if some element x is included in the set, it might make some other element y more likely to be included. As we explain in more detail in Sects. 2 and 4.4, an independent distribution seems to facilitate an efficient membership test, but preclude efficient set enumeration; on the other hand, having more dependence and the right type of dependence can enable efficient set enumeration, but may destroy the efficiency of the membership test. We seek a middle ground by choosing a distribution that has a limited amount of dependence, and the right type of dependence.

We next show how to construct a PRSet scheme that emulates our carefully chosen distribution, and prove the construction secure under our new definitions. Our construction relies on the existence of Privately Puncturable PRFs which can be constructed assuming LWE [7, 9, 12, 14]. Our PRSet construction is remotely inspired by the line of work on designing block ciphers and format preserving encryption from pseudorandom functions [41, 46, 50], but our problem definition and solutions are novel and fundamentally different from prior works.

Finally, we use our PRSet scheme to construct a 2-server preprocessing PIR scheme and prove the PIR scheme correct and secure. Our construction is inspired by Corrigan-Gibbs and Kogan [20] but differs in several important details. The proofs are rather technical and involved. Perhaps somewhat surprisingly, proving correctness turns out to be the most technically challenging part of our proof, although proving privacy is also non-trivial. Our PIR scheme runs k parallel instances of a single-copy PIR scheme. We need to prove occasional correctness of each single-copy scheme, and use majority voting among all instances to amplify correctness. Unfortunately, we cannot easily argue occasional correctness of the single-copy PIR from the occasional correctness of the PRSet scheme. Part of the challenge arises from the fact that conditioning on events that have taken place skews the distribution of the pseudorandom sets, and we need to make an occasional correctness argument even for this skewed distribution (which does not even have a clean and succinct description). At a very high level, to make the argument work, we make an involved stochastic domination argument that effectively shows that conditioning on the events that have taken place will not worsen the probability of certain bad events that could lead to incorrectness. We refer the reader to Sect. 4.4 for more detailed discussions on the technicalities in the proof.

Non-goals and open questions. Previous preprocessing PIR schemes in the unbounded query setting are significantly non-optimal in either online bandwidth or computation. Our work is primarily a theoretical exploration aimed at *bridging the important theoretical gap in our understanding. We do not claim immediate practicality of our scheme.* We believe, however, that achieving asymptotical near optimality represents an important step forward towards eventually having a practical PIR scheme. Specifically, we suggest the following possible future directions towards better concrete performance: 1) the parameters in our current theorems are not tight, therefore concrete security parameterization is a potential improvement; and 2) designing a concretely efficient Privately Puncturable PRF would be critical to concrete performance. For example, instantiations based on other assumptions might be more efficient than the current LWE-based schemes.

Besides improving concrete performance, there are also interesting theoretical open questions. One seemingly challenging question is whether we can asymptotically reduce the client online time—the lower bound by Corrigan-Gibbs and Kogan [20] shows that the server computation (or the combined server-client computation) must be at least \sqrt{n} per query, assuming the client downloads \sqrt{n} information from the server during pre-processing. The known lower bound does not rule out schemes with asymptotically smaller client online time.

2 Strawman Attempts

To understand our ideas, it helps to first illustrate a strawman scheme and see why it fails—the toy scheme below is a variant (and slight simplification) of the elegant 2-server preprocessing PIR scheme by Corrigan-Gibbs and Kogan [20]. This toy scheme is meant for illustrating the “core” of the scheme, and is not concerned about compressing storage or bandwidth.

An Inefficient Toy Scheme: Single-Copy Version

Offline preprocessing. (DB_k denotes the k -th bit of the database)

- Client generates \sqrt{n} sets $S_1, S_2, \dots, S_{\sqrt{n}}$. Each $S_j \subseteq \{0, 1, \dots, n - 1\}$ where $j \in [\sqrt{n}]$ is sampled by including each element $i \in \{0, 1, \dots, n - 1\}$ with independent probability^a $1/\sqrt{n}$.
- Client sends the resulting sets $S_1, \dots, S_{\sqrt{n}}$ to **Left**. For each set $j \in [\sqrt{n}]$, **Left** responds with the parity bit $p_j := \bigoplus_{k \in S_j} DB_k$ of indices in the set.
- Client stores the hint $T := \{T_j := (S_j, p_j)\}_{j \in [\sqrt{n}]}$.

Online query for index $x \in \{0, 1, \dots, n - 1\}$.

- **Query:** (Client \Leftrightarrow Right)
 1. Find an entry $T_j := (S_j, p_j)$ in its hint table T such that $x \in S_j$. Let $S^* := S_j$ if found, else let S^* be a fresh random set containing x .
 2. Send the set $S := \text{Resample}(S^*, x)$ to **Right**, where $\text{Resample}(S^*, x)$ outputs a set almost identical to S^* , except that the coin used to determine x 's membership is re-tossed.
 3. Upon obtaining a response $p := \bigoplus_{k \in S} DB_k$ from **Right**, output the candidate answer $\beta' := p_j \oplus p$ or $\beta' := 0$ if no such T_j was found earlier.
 4. Client obtains the true answer $\beta := DB_x$ —the full scheme will repeat this single-copy scheme k times, and β is computed as a majority vote among the k candidate answers, which is guaranteed to be correct except with negligible probability.
- **Refresh** (Client \Leftrightarrow Left)
 1. Client samples a random set S containing x , and then lets $S' := \text{Resample}(S, x)$, and sends S' to **Left** (notice that this is equivalent to just sampling a fresh set, but we write it this way for later convenience).
 2. **Left** responds with $p := \bigoplus_{k \in S'} DB_k$. If a table entry T_j containing x was found and consumed earlier, **Client** replaces T_j with $(S, p \oplus \beta)$.

^a The work of Corrigan-Gibbs and Kogan [20] samples a set of fixed size \sqrt{n} , whereas in our particular variant, the size of each set is a random variable whose expectation is \sqrt{n} .

In this toy scheme, during pre-processing, the client samples \sqrt{n} sets each containing \sqrt{n} randomly chosen bits, and downloads the parity of each set from

the left server. During an online query, suppose the client wants the index i , it finds a set S^* containing i . It then resamples the decision whether i should belong to the set, and the resampled set S removes i with high probability. It sends the resampled set S to the right server, which returns its parity. Now, if such a set S^* was found, XORing the parity of the set S^* and the set S gives client the correct answer with high probability. To support an unbounded number of queries, the client performs a refresh procedure with the left server to replenish the set that was just consumed.

Correctness amplification through parallel repetition. The above toy scheme guarantees correctness for the query x , provided that 1) an entry $T_j := (S_j, p_j)$ containing x is found, and 2) $\text{Resample}(S_j, x)$ happens to remove x from the set S_j . It is not hard to prove that correctness is guaranteed with probability at least $3/5$ for sufficiently large n . To amplify correctness, we can run k copies of the scheme, and instead of calling the true-answer oracle to obtain the answer β , we set β to be the majority vote among the k candidate answers, which is correct with $1 - 2^{-\Theta(k)}$ probability due to the standard Chernoff bound. If we set $k = \omega(\log \lambda)$, then the failure probability would be negligibly small in λ .

Privacy. In the inefficient toy scheme, left-server privacy is easy to see: basically the left server **Left** sees \sqrt{n} random sets during the offline phase. During each online query, it sees a random set as well.

Arguing right-server privacy is a little more subtle. The right server **Right** is not involved during the offline phase. We want to show that for each online query, **Right** sees a fresh random set. Recall that during a query for x , the client finds an entry $T_j := (S_j, p_j)$ such that $S_j \ni x$. It lets $S^* := S_j$ if such an entry T_j is found, else S^* is a fresh random set containing x . The client now sends $\text{Resample}(S^*, x)$ to **Right** and if such a T_j was found and consumed, it replaces T_j with a fresh set containing x . We can prove right-server privacy by induction: suppose that conditioned on **Right**'s view so far, the client's hint table T contains \sqrt{n} independent random sets (note that this is true at the beginning of the online phase). Then, we can argue that during the next query for x , $\text{Resample}(S^*, x)$ is distributed as a fresh random set conditioned on **Right**'s view so far; and moreover, at the end of the query, the client's hint table T is distributed as \sqrt{n} independent random sets conditioned on **Right**'s view so far.

Performance bounds. In the toy scheme, the bandwidth and server runtime are $O(\sqrt{n})$ for each online query. If the client adopts an efficient data structure for testing set membership, the client's runtime can also be upper bounded by $O(\sqrt{n})$ per query, but its storage is $O(n)$. We want to reduce the online bandwidth to polylogarithmic and reduce the client-side storage to sublinear, while preserving the $\tilde{O}(\sqrt{n})$ online time for both the server and the client.

Strawman ideas for improving efficiency. A failed attempt to improve efficiency is the following. Let us generate each set using a pseudorandom function (PRF) rather than using true randomness. Specifically, we may assume that the $\text{PRF}(\text{sk}, \cdot)$ outputs a number in $[n]$, and an element $i \in \{0, 1, \dots, n - 1\}$ is

considered in the set iff $\text{PRF}(\text{sk}, i) \in [1, \sqrt{n}]$. Moreover, sampling a pseudorandom set would boil down to sampling a fresh PRF secret key.

In this way, a pseudorandom set can be succinctly represented by a PRF secret key, and we can improve the client’s storage to $\sqrt{n} \cdot \chi(\lambda)$ where $\chi(\lambda)$ is an upper bound on the length of the PRF key. During the online phase, the client needs to resample the set at the point x where $x \in \{0, 1, \dots, n - 1\}$ is the current query. If we could represent this locally resampled set succinctly too, then we can reduce the online bandwidth.

To achieve this, our idea is to adopt a Privately Puncturable PRF [7, 9, 14]. A Puncturable PRF is a PRF with the following additional functionality: given a point x and the secret key sk , one can call the $\text{sk}_x \leftarrow \text{Puncture}(\text{sk}, x)$ function to obtain a secret key sk_x that allows one to evaluate the PRF correctly at any point other than x . In an ordinary Puncturable PRF construction [30], using the punctured key sk_x to evaluate over the point x could result in an invalid symbol \perp . In contrast, a *Privately* Puncturable PRF allows one to remove a point x and obtain a punctured key sk_x ; however, the punctured key sk_x does not disclose the point x . For sk_x to hide x , it must be that using sk_x to evaluate over the point x yields a non- \perp outcome r . Not only so, imprecisely speaking, to a computationally bounded adversary, calling $\text{Puncture}(\text{sk}, x)$ should behave just like resampling the PRF’s outcome at the point x .

If we use a Privately Puncturable PRF to construct a pseudorandom set like above, during each online query, we obtain a construct which we call a *Privately Puncturable Pseudorandom Set*. Generating a pseudorandom set is achieved by sampling a PRF key sk . Further, given a set represented by sk that contains a specific element x , one can perform a puncturing operation at x to derive a punctured secret key sk_x —this puncturing procedure acts as if we resampled the coins that determine whether x is in the set or not.

With such a Privately Puncturable Pseudorandom set, during each online query, the client can find a secret key sk from its table T that contains the queried element $x \in \{0, 1, \dots, n - 1\}$ (or sample a random sk containing x if not found), puncture the element x from the set sk , and send the punctured key sk_x to the right server. Similarly, to perform a refresh operation with the left server, the client simply samples a key sk' such that the associated set contains x , puncture x from sk' , and send the resulting punctured key sk'_x to the left server. This approach allows us to compress the online bandwidth to $O(\chi(\lambda))$ bits per copy (and recall that there are $k = \omega(\log \lambda)$ parallel copies), where $\chi(\lambda)$ denotes the length of a punctured key.

Unfortunately, this idea completely fails because to generate the set from a secret key sk , the server would have to do a linear amount of work! This defeats our original goal of achieving sublinear online runtime.

Corrigan-Gibbs and Kogan’s variant and why it fails too. At this point, we also briefly overview the approach of Corrigan-Gibbs and Kogan [20]. They adopt a different PRSet construction that indeed allows efficient set enumeration in roughly \sqrt{n} (rather than linear in n) time. Unfortunately, their scheme does not offer a puncturing procedure that achieves any non-trivial efficiency;

thus in each online query, the client has to send an entire $(\sqrt{n} - 1)$ -sized set (rather than a punctured secret key) to each server. More specifically, Corrigan-Gibbs and Kogan [20] use a Pseudorandom Permutation (PRP) on the domain $\{0, 1, \dots, n - 1\}$ to sample a pseudorandom set. A secret key \mathbf{sk} of a PRP scheme defines a corresponding set $\{\text{PRP}(\mathbf{sk}, i)\}_{i \in \{0, 1, \dots, \sqrt{n} - 1\}}$. Thus, the definition of the set itself gives an efficient set enumeration algorithm. To determine whether an element $x \in \{0, 1, \dots, n - 1\}$ is in the set generated by \mathbf{sk} , simply check whether $\text{PRP}^{-1}(\mathbf{sk}, x) \in \{0, 1, \dots, \sqrt{n} - 1\}$. Their approach samples the set from a different distribution than our earlier strawman—in particular, the sampled set is of fixed size \sqrt{n} , and therefore x being in the set is not independent of whether $y \neq x$ is in the set (even when the PRP is replaced with a completely random permutation). For this reason, during the online phase, they adopt a slightly different approach than our earlier strawman: after finding a set either from the table T or freshly generated that contains the queried element x , they remove x from the set with high probability, but with a small probability, they remove a random element other than x . In this way, the right server sees a random set of size exactly $\sqrt{n} - 1$, and the same applies to the left server.

The main problem with their approach is that it is not amenable to puncturing (with non-trivial efficiency). In fact, Boneh, Kim, and Wu proved the non-existence of Puncturable PRPs [8]. In our case, the domain size n is polynomially bounded, and even if we punctured a point x from the PRP, the adversary can easily recover $\text{PRP}(\mathbf{sk}, x)$ by evaluating $\text{PRP}(\mathbf{sk}, \cdot)$ at all other points.

To get around the non-existence of puncturable PRP barrier, one might be tempted to compute the pseudorandom set as $\{\text{PRF}(\mathbf{sk}, i)\}_{i \in \{0, 1, \dots, \sqrt{n} - 1\}}$ instead, i.e., essentially the “dual” of our earlier PRF-based strawman scheme. While this approach allows for efficient set enumeration, it precludes efficient membership testing which, in our context, would make the client’s online runtime linear.

3 Generalized Privately Puncturable Pseudorandom Set

To summarize the above discussion, we would like to construct a Privately Puncturable Pseudorandom Set (PRSet) scheme with some non-trivial security and efficiency requirements which we shall state shortly after defining the syntax:

- $(\mathbf{sk}, \mathbf{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$: given the security parameter 1^λ and the universe size n , samples a secret key \mathbf{sk} and a corresponding master secret key³ \mathbf{msk} ;
- $S \leftarrow \mathbf{Set}(\mathbf{sk})$: a deterministic algorithm that outputs a set S given the secret key \mathbf{sk} ;
- $b \leftarrow \mathbf{Member}(\mathbf{sk}, x)$: given a secret key \mathbf{sk} and an element $x \in \{0, 1, \dots, n - 1\}$, output a bit indicating whether $x \in \mathbf{Set}(\mathbf{sk})$; and
- $\mathbf{sk}_x \leftarrow \mathbf{Puncture}(\mathbf{msk}, x)$: given a master secret key \mathbf{msk} and an element $x \in \{0, 1, \dots, n - 1\}$, outputs a secret key \mathbf{sk}_x punctured at x .

³ The secret key \mathbf{sk} is needed to enumerate the set, whereas the \mathbf{msk} contains extra secret information needed for computing a punctured key. Jumping ahead, in our PIR scheme, the secret key \mathbf{sk} can be sent to the server whereas the master secret key \mathbf{msk} is kept secret by the client.

We note that a PRSet scheme is parametrized by a family of distributions \mathcal{D}_n . The pseudorandom set generated by the PRSet scheme should emulate the distribution \mathcal{D}_n —we will define this more formally shortly.

Efficiency requirements. Our goal is to use the PRSet scheme to sample pseudorandom sets of size roughly \sqrt{n} . For efficiency, we want that enumerating the set can be accomplished with the **Set(sk)** algorithm, taking time roughly \sqrt{n} (rather than linear in n). Additionally, we want that the membership test algorithm, i.e., **Member(sk, x)**, completes in polylogarithmic time.

3.1 Security Definitions

For security, we want the following:

1. **Pseudorandomness w.r.t. some distribution \mathcal{D}_n :** given a randomly sampled secret key $(\text{sk}, _) \leftarrow \mathbf{Gen}(1^\lambda, n)$, the associated set **Set(sk)** is computationally indistinguishable from a set sampled at random from some distribution \mathcal{D}_n —we shall specify the distribution \mathcal{D}_n later;
2. **Security w.r.t. puncturing I:** we want the following two distributions to be computationally indistinguishable for any $x \in \{0, 1, \dots, n-1\}$:
 - Sample $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$ until **Set(sk)** contains x , and output **Puncture(msk, x)**.
 - Sample $(\text{sk}, _) \leftarrow \mathbf{Gen}(1^\lambda, n)$ and output **sk**.

The above definition says that a key punctured at any point is computationally indistinguishable from an unpunctured key, which implies that a punctured secret key should be simulatable without knowledge of the point x being punctured. In our PIR scheme, we only need the latter property, i.e., that a punctured key is simulatable without knowledge of the point being punctured—but we define this slightly stronger version for simplicity.

3. **Security w.r.t. puncturing II** (defined w.r.t. \mathcal{D}_n): we want the following two distributions to be computationally indistinguishable for any $x \in \{0, 1, \dots, n-1\}$:
 - Sample $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$ until **Set(sk)** contains x , let $\text{sk}_x \leftarrow \mathbf{Puncture}(\text{msk}, x)$, and output $(\mathbf{Set}(\text{sk}), x \in \mathbf{Set}(\text{sk}_x))$ where “ $x \in \mathbf{Set}(\text{sk}_x)$ ” denotes the boolean predicate whether $x \in \mathbf{Set}(\text{sk}_x)$.
 - Sample $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$ until **Set(sk)** contains x , and output $(\mathbf{Set}(\text{sk}), \text{Bernoulli}(\rho))$ where $\rho := \Pr_{S \leftarrow \mathcal{D}_n}[x \in S]$.

Intuitively, the above says that knowing the unpunctured set reveals nothing about whether x still belongs to the set after puncturing x from the set.

Remark 2. Jumping ahead, the “security w.r.t. puncturing I” property will be used in proving the privacy of our PIR scheme, and the “security w.r.t. puncturing II” property will be needed for proving correctness—it turns out that the correctness proof is rather technical (see Sect. 4.4 for further discussions).

3.2 Defining Occasional Correctness

From the strawman attempts described in Sect. 2, we are essentially faced with the following dilemma. Consider some distribution \mathcal{D}_n which the pseudorandom set tries to emulate. On one hand, we want each element to be included in the set with *independent* probability, since this would enable puncturing and efficient membership test. On the other hand, we do not want complete independence among elements, since it would preclude efficient set enumeration. It seems like we have hit a wall, but what comes to our rescue is the observation that our single-copy scheme need not guarantee $(1 - \text{negl}(\lambda))$ -correctness. Since we can take majority vote among $k = \omega(\log \lambda)$ parallel copies, it suffices for each copy to have 2/3-correctness. We therefore hope to seek middle ground between the seemingly conflicting requirements by relaxing correctness.

Informally speaking, we want the following notion of occasional correctness: with $1 - o(1)$ probability over the choice of a PRSet secret key that contains an element $x \in \{0, 1, \dots, n - 1\}$, puncturing at an arbitrary point x would *remove the point x from the set, and only x* . Recall that earlier, we said that puncturing at x should behave as if we resampled the choice whether x is in the set or not, independent of the unpunctured set (see “security w.r.t. puncturing II”). Thus, the relaxed correctness requirement intuitively implies that the resampling that happens at puncturing should only choose to include x in the punctured set with small (but possibly non-negligible) probability. Furthermore, jumping ahead, in our construction, puncturing at x may occasionally end up removing other elements besides x from the set, but this should not happen too often.

It turns out that to formally prove our PIR scheme secure, we actually need a more refined occasional correctness definition. Specifically, our formal definition lets us specify exactly which set of elements are related to x such that they might accidentally get evicted from the set due to the puncturing of x . Further, we also need to define an extra monotonicity condition that the puncturing operation never adds an element to the set.

Formally, we define occasional correctness as below.

Functionality preservation under puncturing. To define functionality preservation, we introduce a symmetric boolean predicate $\text{Related}(x, y) : \{0, 1, \dots, n - 1\}^2 \rightarrow \{0, 1\}$, that outputs whether two elements x and y are related or not. We may assume that $\text{Related}(x, y) = \text{Related}(y, x)$.

We say that $\text{PRSet} := (\mathbf{Gen}, \mathbf{Set}, \mathbf{Member}, \mathbf{Puncture})$ satisfies functionality preservation w.r.t. the Related predicate, iff for any $\lambda, n \in \mathbb{N}$, with probability $1 - \text{negl}(\lambda)$ for some negligible function $\text{negl}(\cdot)$, the following holds: let $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$, then, for any $x \in \mathbf{Set}(\text{sk})$: let $\text{sk}_x \leftarrow \mathbf{Puncture}(\text{msk}, x)$:

1. $\mathbf{Set}(\text{sk}_x) \subseteq \mathbf{Set}(\text{sk})$;
2. $\mathbf{Set}(\text{sk}_x)$ runs in time no more than $\mathbf{Set}(\text{sk})$;
3. for any $y \in \mathbf{Set}(\text{sk}) \setminus \mathbf{Set}(\text{sk}_x)$, it must be that $\text{Related}(x, y) = 1$.

Intuitively, the above requires that puncturing results in a subset of the original set; and the set enumeration time can only reduce once a set has been punctured. Moreover, puncturing x can only cause elements related to x to be removed from

the set. Later on, when we instantiate the distribution $S \stackrel{\$}{\leftarrow} \mathcal{D}_n$ that the PRSet scheme tries to emulate, we shall see that *most elements in the sampled set S likely do not have other related elements in S .*

3.3 Choosing a Sampling Distribution

Recall that each element wants to decide at random whether to be included in the sampled set. Our idea is to allow weak dependence in the coins chosen by different elements. Such weak dependence should be sufficient to allow efficient set enumeration, and yet without destroying efficient membership tests. Of course, we have to pay a price for introducing the weak dependence among elements, and indeed we pay in terms of the correctness of puncturing. In our PRSet scheme, puncturing a secret key msk at a point x may, with some small but non-negligible probability over the choice of msk , not only cause the coins for x to be resampled, but also the coins for some elements other than x . When this happens, puncturing at the point x may end up removing other elements from the set, and possibly lead to an incorrect output in our single-copy PIR.

Even with this high-level intuition, identifying a construction that works is challenging. To this end, our approach is very remotely inspired by the line of work on designing block ciphers and format-preserving encryption [41, 46, 50]. Despite the remote reminiscence, of course, our problem definition and solutions are fundamentally different from block ciphers.

To convey the intuition, let us first describe the distribution our PRSet scheme aims to emulate, assuming the existence of a random oracle⁴ $\text{RO} : \{0, 1\}^* \rightarrow \{0, 1\}$. Suppose we sample an RO at random which will determine a pseudo-random set of expected size roughly $\sqrt{n}/\log^2 n$. To determine if an element $x \in \{0, 1, \dots, n - 1\}$ is in the set associated with RO or not, write x as a $\log n$ -bit string, i.e., $x := \{0, 1\}^{\log n}$. We then say that x is in the set iff using RO to “hash” every sufficiently long suffix of $0^{2 \log \log n} || x$ outputs 1. More formally, set membership of $x \in \{0, 1\}^{\log n}$ is defined with the following algorithm:

1. let $z := 0^B || x$, i.e., prepend $B := \lceil 2 \log \log n \rceil$ number of 0s in front of the string x ;
2. we say that x is in the set iff $\text{RO}(z[i : \cdot]) = 1$ for every $i \in [1, \frac{1}{2} \log n + B]$, where $z[i : \cdot]$ denotes the suffix $z[i : \log n + B]$ starting at the index i . For example, $z[1 : \cdot] = z$, $z[2 : \cdot]$ is the string z removing the first bit, and so on.

Toy example. Figure 1 gives a toy example: suppose that $n = 4$, and thus $B = 2 \log \log n = 2$, and $\frac{1}{2} \log n + B = 3$. Then, the string $x = 10$ is in the set iff $\text{RO}(0010) = \text{RO}(010) = \text{RO}(10) = 1$.

The above sampling distribution has the following properties.

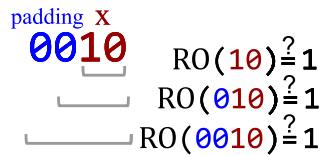


Fig. 1. A toy example.

⁴ Our final scheme does not need any random oracle, the RO is only for exposition.

Expected set size. Each $x \in \{0, 1\}^{\log n}$ is included in the set with probability $2^{-(\frac{1}{2} \log n + B)} \approx 1/(\sqrt{n} \log^2 n)$, and the expected set size is roughly $\sqrt{n}/\log^2 n$.

Fast membership test. The definition itself gives a fast algorithm to test if an element $x \in \{0, 1\}^{\log n}$ is in the set, by making $\frac{1}{2} \log n + B$ calls to RO.

Fast set enumeration. Enumerating all elements in the set can be accomplished by making roughly $\sqrt{n} \cdot \text{poly} \log n$ calls to RO with at least $1 - o(1)$ probability. Let $\ell \geq \frac{1}{2} \log n + 1$, and let Z_ℓ be the set of all strings z of length exactly ℓ , such that using RO to “hash” all suffixes of z of length at least $\frac{1}{2} \log n + 1$ outputs 1. To enumerate the set generated by RO, we can start out $Z_{\frac{1}{2} \log n + 1}$, which takes at most $2^{\frac{1}{2} \log n + 1}$ RO calls to generate. Then, for each $\ell := \frac{1}{2} \log n + 2$ to $\frac{1}{2} \log n + B$, we will generate Z_ℓ from $Z_{\ell-1}$. This can be accomplished by enumerating all elements $z' \in Z_{\ell-1}$, and checking whether $\text{RO}(0||z') = 1$ and $\text{RO}(1||z') = 1$. In our online full version [47], we will prove that with at least $1 - o(1)$ probability, all the Z_ℓ sets encountered along the way will not exceed $\sqrt{n} \cdot \text{poly} \log n$ in size. Thus, with $1 - o(1)$ probability, set enumeration can be accomplished by making at most $\sqrt{n} \cdot \text{poly} \log n$ calls to RO.

Occasional correctness of “puncturing”. Suppose that we sample an RO whose associated set contains the element $x \in \{0, 1\}^{\log n}$. In this idealized world with RO, imagine that puncturing the point x from RO means that we resample the outcomes for $\text{RO}((0^B || x)[i :])$ for every $i \in [1, \frac{1}{2} \log n + B]$. We want to make sure that with $1 - o(1)$ probability over the choice of the RO, puncturing the point x removes x and only x from the resulting set. We prove (a more refined version of) this statement in our online full version [47]. At a high level, to prove this statement, it suffices to prove that the expected number of related elements in the set is $o(1)$, where an element $x' \neq x$ is related to x , iff the longest common suffix of x and x' has length at least $\frac{1}{2} \log n + 1$.

3.4 Our PRSet Scheme

Given the above distribution \mathcal{D}_n , we can derive a PRSet scheme by replacing the RO with a privately puncturable PRF [7, 9, 14]—we review the formal definition for a privately puncturable PRF in the online full version [47]. Puncturing a point $x \in \{0, 1\}^{\log n}$ simply punctures all queries we must make to the PRF to determine x ’s membership. For a punctured key to be indistinguishable from a freshly generated secret key, we puncture a set of “useless” points from a freshly generated secret key as well, since original keys and punctured keys may be trivially distinguishable in the underlying privately puncturable PRF scheme. More formally, let $\text{PRF} := (\text{Gen}, \text{Eval}, \text{Puncture}, \text{PEval})$ be a privately puncturable PRF scheme where **Eval** and **PEval** denote the evaluation algorithms using a normal key and a punctured key, respectively. Our PRSet scheme is described below:

Our PRSet scheme

- **Gen**($1^\lambda, n$): let $B := \lceil 2 \log \log n \rceil$,
 1. call **PRF.Gen** with the appropriate parameters to generate a normal PRF key sk' .
 2. let P be an arbitrary set of $\frac{1}{2} \log n + B$ distinct strings in $\{0, 1\}^{\log n + B}$ that begin with the bit 1;
 3. call $sk \leftarrow \text{PRF.Puncture}(sk', P)$, and output $(sk, msk = sk')$.
- **Set**(sk): similar to the earlier set enumeration algorithm for the distribution \mathcal{D}_n , but replace **RO**(\cdot) calls with calls to **PRF.PEval**(sk, \cdot) instead;
- **Member**(sk, x):
 1. write $x \in \{0, 1\}^{\log n}$ as a binary string, and let $z := 0^B || x$;
 2. if for every $1 \leq i \leq \frac{1}{2} \cdot \log n + B$, $\text{PRF.PEval}(sk, z[i :]) = 1$, then output 1; else output 0.
- **Puncture**(msk, x):
 1. write $x \in \{0, 1\}^{\log n}$ as a binary string, and let $z := 0^B || x$;
 2. let $P := \{z[i :]\}_{i \in [1, \frac{1}{2} \cdot \log n + B]}$ and $sk_P \leftarrow \text{PRF.Puncture}(msk, P)$; output sk_P .

Performance bounds. Our privately puncturable PRF scheme must support puncturing $O(\log n)$ many points. As stated in the online full version [47], we can construct such a privately puncturable PRF with $\tilde{O}(1)$ runtime for **Gen**, **Eval**, and **PEval**, and moreover, each punctured key is of length $\tilde{O}(1)$. Using such a privately puncturable PRF, our resulting PRSet scheme achieves $\tilde{O}(1)$ time for **PRSet.Gen**, **PRSet.Member**, and **PRSet.Puncture** operations, and the expected runtime for **PRSet.Set** is $\tilde{O}(\sqrt{n})$. Bounding the runtime of **PRSet.Set** will require a probabilistic analysis of the distribution \mathcal{D}_n , which we defer to the online full version [47].

We also defer to Sect. 5 a detailed proof of security for our PRSet scheme.

4 Putting it All Together: Our PIR Scheme

4.1 Definitions: Two-Server Preprocessing PIR

In our definition below, the two servers are treated as stateful algorithms **Left** and **Right**, respectively (but in our construction, the only state they need to store is the database itself). The client is treated as a stateful algorithm denoted **Client**. Initially, all of **Client**, **Left**, and **Right** receive the parameters 1^λ and n .

- **Offline setup.** **Client** receives nothing and each of **Left** and **Right** receives the same database $DB \in \{0, 1\}^n$ as input. **Client** sends a single message to **Left**, and **Left** responds with a single message often called a *hint*.
- **Online queries.** The following can be repeated for a priori-unknown polynomially many steps. Upon receiving an index $x \in \{0, 1, \dots, n - 1\}$ to query, **Client** sends a single message to **Left** and a single message to **Right**. It receives

a single response from each server **Left** and **Right**. **Client** then performs some computation and outputs an answer $\beta \in \{0, 1\}$.

Correctness. Given a database $DB \in \{0, 1\}^n$ where the bits are indexed $0, 1, \dots, n - 1$, the correct answer for a query $x \in \{0, 1, \dots, n - 1\}$ is the x -th bit of DB .

For correctness, we require that for any Q, n that are polynomially bounded in λ , there is a negligible function $\text{negl}(\cdot)$, such that for any database $DB \in \{0, 1\}^n$, for any sequence of queries $x_1, x_2, \dots, x_Q \in \{0, 1, \dots, n - 1\}$, an honest execution of the offline/online PIR scheme with DB and queries x_1, x_2, \dots, x_Q returns all correct answers with probability $1 - \text{negl}(\lambda)$.

Privacy. For privacy, we require the following.

- *Left-server privacy.* There is a probabilistic polynomial time (p.p.t.) stateful simulator Sim , such that for any arbitrary (even computationally unbounded) algorithm Right^* , for any non-uniform p.p.t. adversary \mathcal{A} , \mathcal{A} 's views in the **Real** and **Ideal** experiments are computationally indistinguishable:
 1. **Real:** The honest **Client** interacts with \mathcal{A} who acts as the left server and may deviate arbitrarily from the prescribed protocol, and an arbitrary (even computationally unbounded) algorithm Right^* acting as the right server. In every online step t , \mathcal{A} adaptively chooses the next query $x_t \in \{0, 1, \dots, n - 1\}$, and **Client** is invoked with x_t .
 2. **Ideal:** The simulated client Sim interacts with \mathcal{A} who acts as the left server, and an arbitrary (even computationally unbounded) algorithm Right^* acting as the right server. In every online step t , \mathcal{A} adaptively chooses the next query $x_t \in \{0, 1, \dots, n - 1\}$, and Sim is invoked without receiving x_t .
- *Right-server privacy.* Right-server privacy is defined in a symmetric way as above by exchanging left and right.

Intuitively, the above privacy definition requires that any single server alone cannot learn anything about the client's queries; further, this must hold even when both servers can behave maliciously. However, recall that we do not guarantee correctness if one or both server(s) fail to respond correctly.

4.2 Construction

We describe our PIR scheme below, where **Client**, **Left**, **Right** denote the client, the left server, and the right server, respectively.

Our PIR Scheme

Run $k = \omega(\log \lambda)$ parallel copies of the single-copy scheme described below.

Offline setup. For $i = 1$ to $\text{lenT} := 6\sqrt{n} \cdot \log^3 n$ in parallel:

1. **Client:** Sample $(\text{sk}_i, \text{msk}_i) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$, send sk_i to **Left**.
2. **Left:** Run $S_i \leftarrow \text{PRSet.Set}(\text{sk}_i)$. If the runtime of $\text{PRSet.Set}(\text{sk}_i)$, measured in terms of PRF.PEval calls, exceeds $\text{maxT} := 6\sqrt{n} \log^5 n$, return

$p_i := 0$ to Client. Else, return the parity bit $p_i \in \{0, 1\}$ of the set S_i to Client.

3. Client: Save $T_i := (\text{sk}_i, \text{msk}_i, p_i)$ where $T := (T_1, T_2, \dots, T_{\text{len}T})$ denotes a table saved by Client.

Online query for index $x \in \{0, 1, \dots, n - 1\}$.

– **Query** (Client \Leftrightarrow Right):

1. Client:

- (a) Sample

$(\text{sk}, \text{msk}) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}, x) = 1$, append $(\text{sk}, \text{msk}, 0)$ to the end of the table T . (★)

- (b) Henceforth parse $T_i := (\text{sk}_i, \text{msk}_i, p_i)$. Let j be the smallest entry in the table T such that $\text{PRSet.Member}(\text{sk}_j, x) = 1$.

- (c) Call $\tilde{\text{sk}}_j := \text{PRSet.Puncture}(\text{msk}_j, x)$. Send $\tilde{\text{sk}}_j$ to Right.

2. Right: Run $S \leftarrow \text{PRSet.Set}(\tilde{\text{sk}}_j)$. If the runtime exceeds $\text{max}T$, return $p := 0$ to Client. Else, return the parity bit $p \in \{0, 1\}$ of the set S to Client.

3. Client: Let $\beta' := p \oplus p_j$ be a candidate answer of this copy. Let β be the majority vote among the candidate answers of all k copies.

– **Refresh** (Client \Leftrightarrow Left):

1. Client:

- (a) Sample a new $\text{sk}', \text{msk}' \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to the constraint $\text{PRSet.Member}(\text{sk}', x) = 1$. (★)

- (b) Call $\text{sk}'_x \leftarrow \text{PRSet.Puncture}(\text{msk}', x)$, and send sk'_x to Left.

2. Left: Run $S \leftarrow \text{PRSet.Set}(\text{sk}'_x)$. If the runtime exceeds $\text{max}T$, return $p := 0$ to Client. Else, return the parity bit $p \in \{0, 1\}$ of the set S to Client.

3. Client: Replace $T_j := (\text{sk}', \text{msk}', p \oplus \beta)$. Finally, remove the last entry from the table T .

Remark. To obtain *deterministic* performance bounds, we can have the client run the Step 1(a) of both the Query and Refresh phases, marked with (★), at the very beginning of each online query, and simply abort if the number of tries exceeds $\text{max}T$ —in this case, no message is sent and the client outputs a canonical bit 0 as the candidate answer. It is not hard to see that this change does not affect the privacy proof and adds only $o(1)$ correctness failure probability for each instance per query.

Intuitively, the idea here is to summarize the random sets in the earlier toy scheme with the keys of a PRSet scheme, i.e., the client stores the $\text{len}T$ number of keys to represent $\text{len}T$ sets; moreover, the client sends punctured keys to the servers rather than the full sets. By construction, in each copy, the client always obtains answers from the respective servers during the query and refresh phases, but the answers may be incorrect with some small probability. The $k = \omega(\log \lambda)$ parallel repetitions make the overall error probability negligibly small.

Specifically, the answer from the right-server during the query phase may be incorrect if 1) the queried index x is not found in the lenT sets stored by the client; 2) x is found to be in some set represented by $(\text{sk}_j, \text{msk}_j)$, but the parity bit stored by the client is incorrect (see why the refresh phase may cause error shortly); 3) puncturing the key msk_j does not result in exactly the set $\text{Set}(\text{sk}_j) \setminus \{x\}$; or 4) the right server exceeds maxT set enumeration time.

The refresh phase can incur error with small probability too, and thus cause the client to store an incorrect parity bit for the refreshed set. Recall that during refresh, the client computes the parity bit of a newly refreshed set as $\beta + p$ where β is the client’s belief of the answer to the present query, and p is the answer returned by the left server. If either β or p is wrong, the refreshed parity bit may be incorrect. Specifically, p can be wrong if the left server exceeded maxT set enumeration time. Moreover, if the punctured key sk'_x does not give exactly $\text{Set}(\text{sk}') \setminus \{x\}$, then the parity p returned by the left server could be incorrect.

4.3 Performance Analysis

For our performance analysis, we will assume that Step 1(a) of both the Query and Refresh phases, marked (\star) in the PIR scheme, are capped at maxT runtime, since this will give us deterministic performance bounds—see the remark at the end of the PIR algorithm.

We now analyze the performance bounds for each instance of PIR—keep in mind that our final scheme involves $k = \omega(\log \lambda) = \tilde{O}(1)$ parallel instances. Our analysis below also shows how to translate the runtime of the underlying PRSet scheme to the runtime of the resulting PIR scheme. Specifically, we will use our PRSet scheme whose performance bounds are stated in Sect. 3.4.

- *The offline bandwidth and client computation are $\tilde{O}(\sqrt{n})$, the offline server computation is $\tilde{O}(n)$.* The offline client computation is dominated by running PRSet.Gen for $\text{lenT} = \tilde{O}(\sqrt{n})$ number of times; the bandwidth is dominated by transmitting lenT number of PRSet keys to the server and then for the server to transmit 1 parity bit back for each of the lenT keys; and the server computation is dominated by running the PRSet.Set algorithm for lenT number of times, where each PRSet.Set call is capped at $\text{maxT} = \tilde{O}(\sqrt{n})$ runtime.
- *The online server and client runtime is $\tilde{O}(\sqrt{n})$, and the online bandwidth is $\tilde{O}(1)$.* Specifically, during the “Query” phase, the client’s runtime is bounded by the following: Step 1(a) is capped at maxT calls to PRSet.Gen and PRSet.Member; Step 1(b) involves running PRSet.Member at most lenT number of times; the runtime of Step 1(c) and Step 3 is dominated by other steps. During the “Refresh” phase, the client’s runtime involves the following: Step 1(a) is capped at maxT calls to PRSet.Gen and PRSet.Member, and the runtime of Step 1(b) and 3 is dominated by Step 1(a). Both the left and right server’s runtime includes a single call to PRSet.Set capped at maxT , and the cost of computing the parity of at most maxT number of bits. The online bandwidth involves the client sending a single PRSet key to each of the left and right server, and each server sending back one bit.

4.4 Proof Roadmap

Proving our PIR scheme secure turns out to be very much non-trivial. Somewhat surprisingly at first, the most challenging part is actually the proof of occasional correctness of the single-copy version of our PIR scheme (see Sects. 5 and the online full version [47])—even though we are only asking for a relaxed correctness requirement. At a high level, the challenge arises from the fact that the distribution of the PRSet key sk becomes *skewed*, when conditioning on the fact that the key sk is chosen during the online query phase, since it is the first entry in the client’s hint table T that contains the queried element x . In one part of the occasional correctness proof, we need to argue that, imprecisely speaking, despite this skewed distribution, the selected secret key can provide a correct answer to the present query with $1 - o(1)$ probability. In a key technical step, we make a *stochastic domination* type of argument that roughly speaking, proves the following: conditioned on the secret key not having been consumed so far and now being consumed by the present query, it makes it less likely, in comparison with a freshly generated PRSet key, for certain bad events to happen that might lead to incorrectness. To make this argument work, we rewrite the randomized experiment into an equivalent one where the sampling of a subset of the random coins is delayed to the point when they are consumed. In our scheme, multiple bad events can lead to incorrectness of a single copy of the scheme. Therefore, in our proof, we bound the probability of each of these bad events (see the appendices)—to do so, we often view the randomized experiment in different lights, to facilitate the analyses of different bad events.

5 Proofs for Our PRSet Scheme

Lemma 1 (Correctness, pseudorandomness, and functionality preservation under puncturing). *The above PRSet construction satisfies correctness. Further, suppose that the PRF scheme satisfies pseudorandomness; then the PRSet scheme also satisfies pseudorandomness and functionality preservation under puncturing.*

Proof. Correctness follows directly from the construction. Pseudorandomness relies on the pseudorandomness of the PRF through a straightforward reduction. To see functionality preservation, let $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$, let $\text{sk}_x \leftarrow \mathbf{Puncture}(\text{msk}, x)$, and below we may ignore the negligible probability event that the underlying puncturable PRF violates its functionality preservation property. Notice that for every string z that is a suffix of $0^B || x$ of length at least $\frac{1}{2} \log n + 1$, $\text{PRF.PEval}(\text{sk}, z) = 1$, but there may exist such z where $\text{PRF.PEval}(\text{sk}_x, z)$ becomes 0 instead. For any string z that is not a suffix of $0^B || x$ of length at least $\frac{1}{2} \log n + 1$, $\text{PRF.PEval}(\text{sk}, z) = \text{PRF.PEval}(\text{sk}_x, z)$. Given the above observation, “functional preservation under puncturing” is easy to verify.

Lemma 2 (Security w.r.t. puncturing). *Suppose that the PRF scheme satisfies pseudorandomness and privacy w.r.t. puncturing as defined in the online*

full version [47]. Then, the above PRSet construction satisfies security w.r.t. puncturing.

Proof. We need to prove two properties.

First property. We begin by proving the first property, that is, the following distributions are computationally indistinguishable for any $x \in \{0, 1, \dots, n-1\}$:

- Expt_0 : Repeat $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$ until $x \in \mathbf{Set}(\text{sk})$, let $\text{sk}_x \leftarrow \mathbf{Puncture}(\text{msk}, x)$, output sk_x .
- Expt_1 : $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$ and output sk .

We define an intermediate hybrid experiment Hyb : sample $(\text{sk}, \text{msk}) \leftarrow \mathbf{Gen}(1^\lambda, n)$, let $\text{sk}_x \leftarrow \mathbf{Puncture}(\text{msk}, x)$, and output sk_x .

Claim 1. *Suppose that the puncturable PRF satisfies pseudorandomness as defined in the online full version [47]. Then, Expt_0 and Hyb are computationally indistinguishable.*

Proof. Suppose that there is an efficient adversary \mathcal{A} that can distinguish Expt_0 and Hyb with non-negligible probability. We can construct an efficient reduction \mathcal{B} that breaks the pseudorandomness of the PRF scheme.

Let P_x denote the set containing the $m = \frac{1}{2} \log n + B$ queries we need to make to determine whether x is in the set. \mathcal{B} asks its own challenger denoted \mathcal{C} for a PRF key punctured at P_x , and it obtains sk_{P_x} . It forwards sk_{P_x} to \mathcal{A} . \mathcal{B} then obtains a vector of bits denoted $\beta := (\beta_1, \dots, \beta_m)$ as the purported outcomes for $\{\mathbf{Eval}(\text{sk}, y)\}_{y \in P_x}$. If $\beta = \mathbf{1}$, then \mathcal{B} outputs whatever \mathcal{A} outputs. Else, it outputs a random bit.

Case 1. If the challenger \mathcal{C} is using real values for $\{\mathbf{Eval}(\text{sk}, y)\}_{y \in P_x}$, then \mathcal{A} 's view is identically distributed as Expt_0 . The probability that \mathcal{B} outputs 1 is

$$p := \Pr[\mathcal{A}(\text{Expt}_0) = 1] \cdot \Pr[\beta = \mathbf{1}] + \frac{1}{2} \cdot \Pr[\beta \neq \mathbf{1}]$$

Case 2. If the challenger \mathcal{C} is using random values in place of $\{\mathbf{Eval}(\text{sk}, y)\}_{y \in P_x}$, then \mathcal{A} 's view is identically distributed as Hyb . In this case, the probability that \mathcal{B} outputs 1 is equal to

$$p' := \Pr[\mathcal{A}(\text{Hyb}) = 1] \cdot \Pr[\beta = \mathbf{1}] + \frac{1}{2} \cdot \Pr[\beta \neq \mathbf{1}]$$

Note that in Case 1, $|\Pr[\beta = \mathbf{1}] - 1/2^m| \leq \text{negl}(\lambda)$ due to the pseudorandomness of the PRF; and in Case 2 $\Pr[\beta = \mathbf{1}] = 1/2^m$. Moreover, $1/2^m$ is non-negligible due to the choice of m . Therefore, if $|\Pr[\mathcal{A}(\text{Expt}_0) = 1] - \Pr[\mathcal{A}(\text{Hyb}) = 1]|$ is non-negligible, then $|p - p'|$ would be non-negligible, too.

Claim 2. *Suppose that the puncturable PRF satisfies privacy w.r.t. puncturing as defined in the online full version [47]. Then, Hyb is computationally indistinguishable from Expt_1 .*

Proof. Follows from a straightforward reduction to the privacy w.r.t. puncturing property of the PRF.

The computational indistinguishability of Expt_0 and Expt_1 now follows from Claim 1 and Claim 2.

Second property. We next prove the second property, that is, we want to show that the following two distributions are computationally indistinguishable:

- Expt_0^* : Repeat $(\text{sk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, n)$ until $x \in \text{Set}(\text{sk})$, let $\text{sk}_x \leftarrow \text{Puncture}(\text{msk}, x)$, and output $(\text{Set}(\text{sk}), x \in \text{Set}(\text{sk}_x))$ where $x \in \text{Set}(\text{sk}_x)$ denotes a boolean predicate.
- Expt_1^* : Repeat $(\text{sk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, n)$ until $x \in \text{Set}(\text{sk})$, and output $(\text{Set}(\text{sk}), \text{Bernoulli}(\rho))$ where $\rho := 2^{-(\frac{1}{2} \log n + B)}$.

Let $(\text{sk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, n)$ until $x \in \text{Set}(\text{sk})$, and let $\text{sk}_x \leftarrow \text{Puncture}(\text{msk}, x)$. Observe that there is a deterministic, polynomial-time function **Reconstruct** such that $\text{Reconstruct}(\text{sk}_x, x) = \text{Set}(\text{sk})$. Essentially, **Reconstruct** uses answers to $\text{PRF.PEval}(\text{sk}_x, \cdot)$ calls to determine set membership, except that when encountering any string z that is a suffix of $0^B || x$ of length at least $\frac{1}{2} \log n + 1$, override the outcome of $\text{PRF.PEval}(\text{sk}_x, z)$ to 1.

We can therefore rewrite Expt_0^* as the following experiment **Hyb**: repeat $(\text{sk}, \text{msk}) \leftarrow \text{Gen}(1^\lambda, n)$ until $x \in \text{Set}(\text{sk})$, let $\text{sk}_x \leftarrow \text{Puncture}(\text{msk}, x)$, and output $(\text{Reconstruct}(\text{sk}_x, x), x \in \text{Set}(\text{sk}_x))$.

Due to the first property which we just proved, the above distribution **Hyb** is computationally indistinguishable from the following **Hyb'**: $(\text{sk}, \cdot) \leftarrow \text{Gen}(1^\lambda, n)$, and output $(\text{Reconstruct}(\text{sk}, x), x \in \text{Set}(\text{sk}))$.

Now, consider the experiment **Ideal** which is defined just like in **Hyb**, except that sampling a PRF secret key is replaced with sampling an RO, and to determine set membership, any call to $\text{PRF.PEval}(\text{sk}, \cdot)$ is replaced with $\text{RO}(\cdot)$. In **Ideal**, $\text{Reconstruct}(\text{RO}, x)$ does not need to look at the coins that determine x 's membership in the set. Based on this observation as well as the pseudo-randomness of the underlying PRF, we conclude that **Ideal** is computationally indistinguishable from Expt_1^* .

Deferred contents. We defer the probabilistic analysis of the distribution \mathcal{D}_n , and proofs for the runtime of set enumeration to the online full version [47].

6 Proofs for Our PIR Scheme

Single-copy variant of our PIR scheme. In our proofs, we consider a single-copy variant of our PIR scheme. In the single-copy scheme, we set the number of parallel instances $k := 1$. Further, we imagine that the true answer β is obtained from some true-answer oracle rather than taking majority vote.

6.1 Privacy Proof

We focus on the single-copy variant, and prove its privacy.

Theorem 2 (Left-server privacy). *Suppose that the PRSet scheme satisfies (the first property in) “security w.r.t. puncturing”. Then, the single-copy scheme satisfies left-server privacy.*

Proof. We define the following simulator Sim which fully specifies the ideal experiment Ideal:

- *Offline setup.* For $i = 1$ to $\text{lenT} := 6\sqrt{n} \cdot \log^3 n$: sample $(\text{sk}_i, \text{msk}_i) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ and send $\{\text{sk}_i\}_{i \in [1, \text{lenT}]}$ to \mathcal{A} acting as the left server.
- *Online queries.* For each online query, sample $(\text{sk}', \text{msk}') \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ and send sk' to \mathcal{A} acting as the left server.

The computational indistinguishability of \mathcal{A} 's views in Real and Ideal follow due to a straightforward hybrid argument relying on the “security w.r.t. puncturing” property of the PRSet scheme. Specifically, let Q be the total number of queries in the online phase. We define a sequence of hybrid experiments $\{\text{Hyb}_i\}_{i \in \{0, 1, \dots, Q\}}$, where in Hyb_i , during the first i online steps, \mathcal{A} (acting as the left server) receives a message constructed like in Ideal, and during the remaining $Q - i$ online steps, \mathcal{A} receives a message constructed like in Real. Clearly, $\text{Hyb}_0 = \text{Real}$ and $\text{Hyb}_Q = \text{Ideal}$. It suffices to show that any two adjacent hybrid experiments are computationally indistinguishable, and this follows due to a straightforward reduction to the “security w.r.t. puncturing” property of the PRSet scheme.

Theorem 3 (Right-server privacy). *Suppose that the PRSet scheme satisfies (the first property in) security w.r.t. puncturing. Then, the single-copy scheme satisfies right-server privacy.*

Proof. We define the following simulator Sim which fully specifies the ideal experiment Ideal:

- *Offline setup.* \mathcal{A} , acting as the right server, receives nothing.
- *Online queries.* For each online query, sample $(\text{sk}', \text{msk}') \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ and send sk' to \mathcal{A} acting as the right server.

We now need to argue that any non-uniform p.p.t. \mathcal{A} 's views in Real and Ideal are computationally indistinguishable.

Real*. First, we consider the following experiment Real*.

- *Offline setup.* For each $i \in [1, \text{lenT}]$, Client samples $(\text{sk}_i, \text{msk}_i) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$, and lets $T_i := (\text{sk}_i, \text{msk}_i)$. The adversary \mathcal{A} , acting as the right server, receives nothing.
- *Online queries.* For each online query $x \in \{0, 1, \dots, n - 1\}$:
 - a) Client samples $(\text{sk}, \text{msk}) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}, x) = 1$, and appends (sk, msk) to the end of the table T .

- b) Client finds the smallest entry $T_j := (\text{sk}_j, \text{msk}_j)$ in T such that $\text{PRSet.Member}(\text{sk}_j, x) = 1$. It sends $\text{PRSet.Puncture}(\text{msk}_j, x)$ to \mathcal{A} acting as the right server.
- c) Client samples $(\text{sk}', \text{msk}') \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}, x) = 1$, overwrites T_j with $(\text{sk}', \text{msk}')$, and removes the last entry from T .

Real^* is just a rewrite of Real throwing away terms that we do not care. Let $\text{View}^{\text{Real}}$ and $\text{View}^{\text{Real}^*}$ denote \mathcal{A} 's view and the client's table T (truncating the third field of each entry in Real) at the beginning of each online query, in the experiments Real and Real^* , respectively. We have that even for a computationally unbounded \mathcal{A} , $\text{View}^{\text{Real}}$ and $\text{View}^{\text{Real}^*}$ are identically distributed.

Fact 1. *In Real^* , for every online step t , even if \mathcal{A} is computationally unbounded, and even when conditioned on \mathcal{A} 's view over the first $t - 1$ steps,*

- let $x \in \{0, 1, \dots, n - 1\}$ be the t -th online query, the message \mathcal{A} receives in the t -th query is distributed as: sample $(\text{sk}, \text{msk}) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}, x) = 1$ and output $\text{PRSet.Puncture}(\text{sk}, x)$;
- at the end of the t -th online query, the client's table T is a fresh uniform sample from $\text{PRSet.Gen}(1^\lambda, n)^{\text{len}T}$ independent of the message \mathcal{A} receives during the t -th query, i.e., T contains a sample of $\text{len}T$ uniform, independent entries from the distribution $\text{PRSet.Gen}(1^\lambda, n)$.

Proof. We can prove by induction.

Base case. At the end of the offline phase (henceforth also called the 0-th query), indeed the client's table T is a uniform sample from the distribution $\text{PRSet.Gen}(1^\lambda, n)^{\text{len}T}$.

Inductive step. Suppose that at the end of the t -th step, the client's table T is uniform sample from the distribution $\text{PRSet.Gen}(1^\lambda, n)^{\text{len}T}$ even when conditioned on \mathcal{A} 's view in the first t steps. We now prove that the stated claims hold for $t + 1$. Let $x \in \{0, 1, \dots, n - 1\}$ be the query made in online step $t + 1$, the choice of x depends only on \mathcal{A} 's view in the first t online queries. Henceforth, for $i \in [1, \text{len}T]$, let $\alpha_{i,x}$ be the probability that in a random sample from the distribution $\text{PRSet.Gen}(1^\lambda, n)^{\text{len}T}$, the first entry that contains x is i . Let $\alpha_{\text{len}T+1,x} := 1 - \sum_{i=1}^{\text{len}T} \alpha_{i,x}$.

Consider the following experiment **Expt**:

- Client samples an index $u \in [\text{len}T + 1]$ such that $u = i$ with probability $\alpha_{i,x}$.
- $\forall j < u$, Client samples $T_j := (\text{sk}_j, \text{msk}_j) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}_j, x) = 0$.
- For u , Client samples (sk, msk) and $(\text{sk}', \text{msk}')$ independently from the distribution $\text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}_j, x) = 1$. It sends $\text{PRSet.Puncture}(\text{sk}, x)$ to \mathcal{A} and saves $T_u := (\text{sk}', \text{msk}')$.
- $\forall j \in [u + 1, \text{len}T + 1]$, Client samples $T_j := (\text{sk}_j, \text{msk}_j) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$.
- Finally, Client removes last entry from T .

Let $\text{View}_{t+1}^{\text{Real}^*}$ be the message \mathcal{A} receives during the $(t+1)$ -st query as well as the client’s table T at the end of the $(t+1)$ -st query in Real^* . Let $\text{View}^{\text{Expt}}$ be the message \mathcal{A} receives as well as the client’s table T at the end in Expt . Suppose that the induction hypothesis holds, then it is not hard to see that $\text{View}^{\text{Expt}}$ is identically distributed as $\text{View}_{t+1}^{\text{Real}^*}$ even when conditioning on the view of \mathcal{A} in the first t queries in Real^* , and even when \mathcal{A} is computationally unbounded.

In the experiment Expt , it is not hard to see the distribution $\text{View}^{\text{Expt}}$ is the following: T is sampled at random from $\text{PRSet.Gen}(1^\lambda, n)^{\text{len}T}$, and \mathcal{A} ’s received message is distributed as: sample $(\text{sk}, \text{msk}) \leftarrow \text{PRSet.Gen}(1^\lambda, n)$ subject to $\text{PRSet.Member}(\text{sk}, x) = 1$ and output $\text{PRSet.Puncture}(\text{sk}, x)$.

Given Fact 1, we can prove that any non-uniform p.p.t. \mathcal{A} ’s views in Ideal and Real^* are computationally indistinguishable through a standard hybrid argument, relying on the “security w.r.t. puncturing” property of the PRSet scheme—the hybrid sequence is similar to the proof of Theorem 2.

6.2 Correctness Proof

Deferred to the online full version [47].

7 Additional Related Work

Beimel et al. [6] proved that in the original formulation of PIR, the servers must collectively probe all n bits of the database on average to respond to a client’s query. Various techniques have been suggested to overcome this key performance bottleneck, e.g., encoding the server-side database, storing per-client or even per-query server state, batching, introducing assumptions like Virtual-Blackbox obfuscation which is known to be impossible [5], or having many non-colluding servers. We review this line of work below.

As mentioned in Sect. 1, the work of Beimel et al. achieves sublinear online computation by encoding the database into a $n^{1+\epsilon}$ to $\text{poly}(n)$ -sized string. The recent (designated-client) doubly efficient PIR schemes [11, 15] rely on encoding the database as well as having the server store $\Omega(n)$ state per client, which is a significant barrier towards practicality in our motivating applications. Boyle et al. [11] show that assuming Virtual-Blackbox Obfuscation which is known to be impossible [5] (and additional non-standard assumptions that are not yet well understood), one can indeed construct a preprocessing PIR with n^ϵ online runtime and bandwidth, without having to store per-client state at the server.

A related notion called *private anonymous data access* (PANDA) was recently introduced by Hamlin et al. [33]. PANDA is a form of preprocessing PIR which requires a *third-party trusted setup* besides the client and the servers (which is not necessary in our work); and moreover, the server storage and time grow w.r.t. the number of corrupt clients. In our motivating examples, the number of clients is essentially unbounded which makes known PANDA schemes unsuitable. Some works [6, 23] suggested having the server store *per-query* state to reduce

the online time. Specifically, the construction by Beimel et al. [6] requires a linear amount of server storage per query, and this is even worse than per-client storage. Other works [43] improve the online time by making the number of public-key operations sublinear, along with a still *linear* number of symmetric-key operations. Sharding has also been suggested to spread out the server work online [21] but the total work across all servers is still linear.

A couple of works [38, 44] construct preprocessing PIR schemes whose online runtime is marginally sublinear, e.g., roughly $O(n/\log n)$; and the complexity of these protocols is much larger than Corrigan-Gibbs and Kogan [20].

An elegant line of work suggested batching queries from the same client [3, 4, 32, 34] or among multiple clients [6, 35, 40] to amortize the linear server work among the batch. Our formulation can be viewed as a generalization of batched PIR, since we do not require the requests to come in a batch, and we can nonetheless achieve small online bandwidth and work. The work by Beimel et al. [6] also showed how to get a preprocessing PIR with polylogarithmic online bandwidth and cost assuming polylogarithmically many non-colluding servers, and $\text{poly}(n)$ server space. Toledo et al. [51] consider how to relax the security definition and achieve differential-privacy-style security, to improve the server time to sublinear.

The concurrent work of Kogan and Corrigan-Gibbs [36] gives a practical instantiation of their earlier work [20], with a clever trick to remove the k -fold parallel repetition. Their implementation is indeed in the unbounded query setting. For their particular application, i.e., private blocklist, it turns out that the database is somewhat small, and therefore, they are willing to incur $\Theta(n)$ computation per online query, in exchange for roughly $O(\sqrt{n})$ online time and logarithmic bandwidth. While their implementation is indeed a practical sweet spot for the private blocklist application, for larger databases, incurring linear client time per online query could be prohibitive. Their trick to remove the k -fold repetition does not seem to immediately apply to our construction because we have an additional source of error from our underlying PRSet scheme.

Deferred Contents

In the interest of space, we defer additional details and proofs to the online full version [47].

Acknowledgment. This work is in part supported by a DARPA SIEVE grant under a subcontract from SRI, an ONR YIP award, a Packard Fellowship, a JP Morgan Award, and NSF grants under the award numbers 2001026, 1901047, and 1763742. The authors would like to acknowledge Dima Kogan and Feng-Hao Liu for helpful discussions, and thank the anonymous reviewers for the detailed and thoughtful comments.

References

1. Oblivious DNS over HTTPS. <https://tools.ietf.org/html/draft-pauly-dprive-oblivious-doh-04>

2. Private information retrieval with sublinear online time. Talk by Dmitry Kogan at Charles River Crypto Day (2021)
3. Ali, A., et al.: Communication-computation trade-offs in PIR. Cryptology ePrint Archive, Report 2019/1483 (2019). <https://eprint.iacr.org/2019/1483>
4. Angel, S., Chen, H., Laine, K., Setty, S.T.V.: PIR with compressed queries and amortized query processing. In: S&P (2018)
5. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
6. Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers computation in private information retrieval: PIR with preprocessing. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 55–73. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_4
7. Boneh, D., Kim, S., Montgomery, H.: Private puncturable PRFs from standard lattice assumptions. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 415–445. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_15
8. Boneh, D., Kim, S., Wu, D.J.: Constrained keys for invertible pseudorandom functions. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 237–263. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_9
9. Boneh, D., Lewi, K., Wu, D.J.: Constraining pseudorandom functions privately. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 494–524. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_17
10. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: improvements and extensions. In: CCS (2016)
11. Boyle, E., Ishai, Y., Pass, R., Wootters, M.: Can we access a database both locally and privately? In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 662–693. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_22
12. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and More) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 264–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_10
13. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_28
14. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 446–476. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_16
15. Canetti, R., Holmgren, J., Richelson, S.: Towards doubly efficient private information retrieval. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 694–726. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_23
16. Chang, Y.-C.: Single database private information retrieval with logarithmic communication. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 50–61. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_5
17. Chor, B., Gilboa, N.: Computationally private information retrieval. In: STOC (1997)
18. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS (1995)

19. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* **45**(6), 965–981 (1998)
20. Corrigan-Gibbs, H., Kogan, D.: Private information retrieval with sublinear online time. In: Canteaut, A., Ishai, Y. (eds.) *EUROCRYPT 2020*. LNCS, vol. 12105, pp. 44–75. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_3
21. Demmler, D., Herzberg, A., Schneider, T.: RAID-PIR: practical multi-server PIR. In: *CCSW* (2014)
22. Devadas, S., van Dijk, M., Fletcher, C.W., Ren, L., Shi, E., Wichs, D.: Onion ORAM: a constant bandwidth blowup oblivious RAM. In: *TCC* (2016)
23. Di-Crescenzo, G., Ishai, Y., Ostrovsky, R.: Universal service-providers for database private information retrieval. In: *PODC* (1998)
24. Dvir, Z., Gopi, S.: 2-server PIR with subpolynomial communication. *J. ACM* **63**(4), 1–15 (2016)
25. Garg, S., Mohassel, P., Papamanthou, C.: TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In: Robshaw, M., Katz, J. (eds.) *CRYPTO 2016*. LNCS, vol. 9816, pp. 563–592. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_20
26. Gasarch, W.I.: A survey on private information retrieval. *Bull. EATCS* **82**, 72–107 (2004)
27. Gentry, C., Halevi, S., Lu, S., Ostrovsky, R., Raykova, M., Wichs, D.: Garbled RAM revisited. In: Nguyen, P.Q., Oswald, E. (eds.) *EUROCRYPT 2014*. LNCS, vol. 8441, pp. 405–422. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_23
28. Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) *ICALP 2005*. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_65
29. Goldreich, O.: Towards a theory of software protection and simulation by oblivious RAMs. In: *STOC* (1987)
30. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
31. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. *J. ACM* **43**, 431–473 (1996)
32. Hafiz, S.M., Henry, R.: Querying for queries: indexes of queries for efficient and expressive IT-PIR. In: *CCS* (2017)
33. Hamlin, A., Ostrovsky, R., Weiss, M., Wichs, D.: Private anonymous data access. In: Ishai, Y., Rijmen, V. (eds.) *EUROCRYPT 2019*. LNCS, vol. 11477, pp. 244–273. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_9
34. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Batch codes and their applications. In: *STOC* (2004)
35. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography from anonymity. In: *FOCS*, pp. 239–248 (2006)
36. Kogan, D., Corrigan-Gibbs, H.: Private blacklist lookups with checklist. In: *Usenix Security* (2021)
37. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: *FOCS* (1997)
38. Lipmaa, H.: First CPIR protocol with data-dependent computation. In: Lee, D., Hong, S. (eds.) *ICISC 2009*. LNCS, vol. 5984, pp. 193–210. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14423-3_14

39. Lu, S., Ostrovsky, R.: How to garble RAM programs? In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 719–734. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_42
40. Lueks, W., Goldberg, I.: Sublinear scaling for multi-client private information retrieval. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 168–186. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47854-7_10
41. Morris, B., Rogaway, P.: Sometimes-recurse shuffle - almost-random permutations in logarithmic expected time. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 311–326. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_18
42. Ostrovsky, R., Skeith, W.E.: A survey of single-database private information retrieval: techniques and applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_26
43. Patel, S., Persiano, G., Yeo, K.: Private stateful information retrieval. In: CCS (2018)
44. Pudlák, P., Rödl, V.: Modified ranks of tensors and the size of circuits. In: STOC (1993)
45. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56**(6), 1–40 (2009)
46. Ristenpart, T., Yilek, S.: The mix-and-cut shuffle: small-domain encryption secure against N queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 392–409. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_22
47. Shi, E., Aqeel, W., Chandrasekaran, B., Maggs, B.: Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. Online full version of this paper. Cryptology ePrint Archive, Report 2020/1592 (2020). <https://eprint.iacr.org/2020/1592>
48. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_11
49. Singanamalla, S.: Oblivious DNS over HTTPS (ODOH): a practical privacy enhancement to DNS (2020)
50. Stefanov, E., Shi, E.: FastPRP: fast pseudo-random permutations for small domains. IACR Cryptology ePrint Archive 2012:254 (2012)
51. Toledo, R.R., Danezis, G., Goldberg, I.: Lower-cost ϵ -private information retrieval. In: PETS (2016)
52. Wang, F., Yun, C., Goldwasser, S., Vaikuntanathan, V., Zaharia, M.: Splinter: practical private queries on public data. In: NSDI (2017)
53. Yao, A.C.-C.: Coherent functions and program checkers. In: STOC (1990)



Authenticated Key Exchange and Signatures with Tight Security in the Standard Model

Shuai Han^{1,2}, Tibor Jäger³, Eike Kiltz⁴, Shengli Liu^{1,2,5} (✉),
Jiaxin Pan⁶, Doreen Riepel⁴, and Sven Schäge⁴

¹ School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{dalen17, slliu}@sjtu.edu.cn

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³ Bergische Universität Wuppertal, Wuppertal, Germany
tiber.jager@uni-wuppertal.de

⁴ Ruhr-Universität Bochum, Bochum, Germany

{eike.kiltz, doreen.riepel, sven.schaege}@rub.de

⁵ Westone Cryptologic Research Center, Beijing 100070, China

⁶ Department of Mathematical Sciences,

NTNU – Norwegian University of Science and Technology, Trondheim, Norway
jiaxin.pan@ntnu.no

Abstract. We construct the first authenticated key exchange protocols that achieve tight security in the *standard model*. Previous works either relied on techniques that seem to inherently require a random oracle, or achieved only “Multi-Bit-Guess” security, which is not known to compose tightly, for instance, to build a secure channel.

Our constructions are generic, based on digital signatures and key encapsulation mechanisms (KEMs). The main technical challenges we resolve is to determine suitable KEM security notions which on the one hand are strong enough to yield tight security, but at the same time weak enough to be efficiently instantiable in the standard model, based on standard techniques such as universal hash proof systems.

Digital signature schemes with tight multi-user security in presence of adaptive corruptions are a central building block, which is used in all known constructions of tightly-secure AKE with full forward security. We identify a subtle gap in the security proof of the only previously known efficient standard model scheme by Bader *et al.* (TCC 2015). We develop a new variant, which yields the currently most efficient signature scheme that achieves this strong security notion without random oracles and based on standard hardness assumptions.

Keywords: Authenticated key exchange · Digital signatures · Tightness

1 Introduction

A *tight* security proof establishes a close relation between the security of a cryptosystem and its underlying building blocks, *independent* of deployment parameters such as the number of users, protocol sessions, issued signatures, etc. This enables a theoretically-sound instantiation with optimal parameters, without the need to compensate a security loss by increasing key lengths or group sizes.

AKE. Authenticated key exchange (AKE) protocols enable two parties to authenticate each other and compute a shared session key. In comparison to many other cryptographic primitives, standard security models for AKE are extremely complex. Following the approach of Bellare-Rogaway [5] and Canetti-Krawczyk [7], a very strong active adversary is considered, which essentially “carries” all protocol messages between parties running the protocol and thus is able to modify, replace, replay, drop, or inject arbitrary messages. Furthermore, the adversary may adaptively corrupt parties and reveal session keys while adaptively choosing which session(s) to “attack”.

Achieving security in such a strong and complex model gives very strong security guarantees, but it also makes *tightness* particularly difficult to achieve. Indeed, most security proofs of AKE protocols are extremely lossy, often even with a *quadratic* security loss in the total number of sessions established over the entire lifetime of the protocol. Considering for instance the huge number of TLS connections per day in practice, this loss may be too large to compensate in practice because the resulting increase of deployment parameters would incur an intolerable performance overhead. Hence, such protocols could not be instantiated in a theoretically-sound way.

Therefore tight security of AKE protocols is a well-established research area, with several known constructions [2, 11, 13, 19, 23, 29]. As recently pointed out by Jager et al. [23], some of these constructions [2, 19, 29] consider a “*Multi-Bit-Guess*” (MBG) security experiment, which is not known to compose tightly with primitives that apply the shared session key, e.g., to build a secure channel. The standard and well established security notion in the context of multiple challenges is “*Single-Bit Guess*” (SBG) security. Unfortunately, the only known constructions in the SBG model [11, 13, 23] apply proof techniques that seem to inherently require the random oracle model [4]. For instance, [23] is based on non-committing encryption, which is known to be not instantiable without random oracles [32], whereas [11, 13] use a similar approach based on adaptive reprogramming of the random oracle.

Currently, there exists no AKE protocol which achieves tight security in a standard (SBG) AKE security model, with a security proof in the standard model, without random oracles, not even an impractical one.

DIGITAL SIGNATURES. Digital signatures are a foundational cryptographic primitive and often used to build AKE protocols. All known tightly-secure AKE protocols with full forward security [2, 11, 13, 19, 23, 29] are based on signatures that provide tight existential unforgeability under chosen-message attacks (EUF-CMA), but in a *multi-user* setting and in the presence of an adversary that may *adaptively*

corrupt users to obtain their secret keys (MU-EUF-CMA^{corr} security [2]). It is easy to prove that MU-EUF-CMA^{corr} security is non-tightly implied by standard EUF-CMA security, but with a linear security loss in the number of users.

The construction of a *tightly* MU-EUF-CMA^{corr} secure signature scheme has to overcome the following, seemingly paradoxical technical problem. On the one hand, the reduction must be able to output user secret keys to the adversary, to respond to adaptive secret key corruption queries. However, it cannot apply a guessing argument, as this would incur a tightness loss. Therefore it is forced to “know” the secret keys of *all* users. On the other hand, it must be able to extract a solution to a computationally hard problem from a forgery produced by an adversary. This seems to be in conflict with the fact that the reduction has to know secret keys for all users, as knowledge of the secret key should enable the reduction to compute a “forged” signature by itself, without the adversary. In fact, tight multi-user security is known to be impossible for many signature schemes, for example when the public key uniquely defines the matching secret key [3].

Several previous works have developed techniques to overcome this seeming paradox [1, 2, 12, 19]. Essentially, their approach is to build schemes where secret keys are not uniquely determined by public parameters, along with a reduction that exploits this to evade the paradox. However, all currently known constructions either use the random oracle model, and therefore cannot be used to build tightly-secure AKE in the standard model, or are based on tree-based signatures [2], which yields signatures with hundreds of group elements and thus would incur even more overhead than compensating the security loss with larger parameters. Jumping slightly ahead, we remark that [2] also describes a pairing-based signature scheme with short constant-size signatures, but we identify a gap in the security proof. Hence, currently there is no practical signature scheme which achieves tight security in the multi-user setting with adaptive corruptions.

1.1 Contributions

Summarizing the previous paragraphs, we can formulate the following natural questions related to AKE and signatures:

Do there exist efficient AKEs and signature schemes with tight multi-user security in the standard model?

TIGHTLY-SECURE SIGNATURES. We identify a subtle gap in the MU-EUF-CMA^{corr} security proof of the scheme from [2] with constant-size signatures (namely, SIG_C in [2, Section 2.3]). We did not find a way to close this gap and therefore develop a new variant of this scheme and prove tight MU-EUF-CMA^{corr} security in the standard model. More precisely, SIG_C follows the blueprint of the Blazy-Kiltz-Pan (BKP) identity-based encryption scheme [6], and transforms an algebraic message authentication code (MAC) scheme into a signature scheme with pairings. If the MAC is tightly-secure in a model with adaptive corruptions, so is the signature scheme. We notice, however, that their MAC does not achieve tight security with adaptive corruptions since the corruption queries leak too much secret information to the adversary.

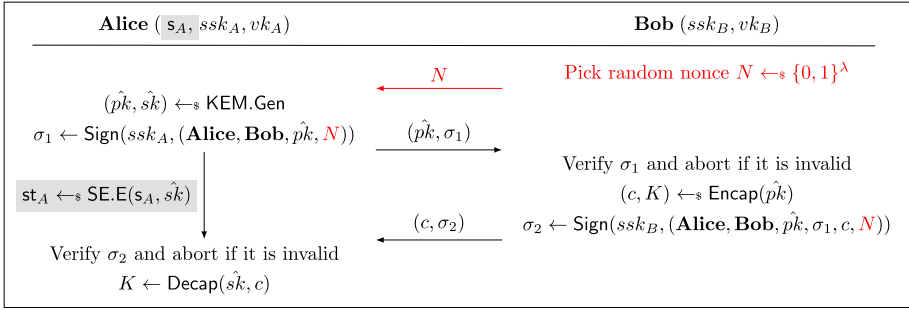


Fig. 1. The two-message protocol $\text{AKE}_{2\text{msg}}$ using the “KEM+2×SIG” approach and the three-message protocols $\text{AKE}_{3\text{msg}}$ (including the red parts) and $\text{AKE}_{3\text{msg}}^{\text{state}}$ (including the red and gray parts) using the “Nonce+KEM+2×SIG” approach. ($\text{AKE}_{3\text{msg}}^{\text{state}}$ additionally uses a symmetric encryption scheme SE.)

To overcome this issue, we borrow recent techniques from tightly-secure hierarchical identity-based encryption schemes [26,27] to construct a new MAC scheme that can be proven tightly secure under adaptive corruptions. Our construction is based on pairings and general random self-reducible matrix Diffie-Hellman (MDDH) assumptions [15]. When instantiated based on the \mathcal{D}_k -MDDH assumption (e.g., k -Lin), a signature consists of $4k + 1$ group elements. That is 5 group elements for $k = 1$ (SXDH). This yields the first tightly MU-EUF-CMA^{corr}-secure signature in the standard model with practical efficiency.

We remark that our new signature scheme circumvents known impossibility results for signatures and MACs [3,30], since these apply only to schemes with re-randomizable signatures or re-randomizable secret keys [3], or deterministic schemes [30]. Our construction is probabilistic and not efficiently re-randomizable in the sense of [3].¹

TIGHTLY-SECURE AKE IN THE STANDARD MODEL. The classical “key encapsulation plus digital signatures” (KEM + 2 × SIG) paradigm to construct AKE protocols gives rise to efficient protocols and is the basis of many constructions, e.g., [7,10,11,13,19,23,29]. To establish a session key, two parties Alice and Bob proceed as follows (cf. Fig. 1). Alice generates an ephemeral KEM key pair (\hat{pk}, \hat{sk}) and sends the signed public key to Bob. Bob then uses this public key to encapsulate a session key, signs the ciphertext, and sends it back to Alice. Alice then obtains the session key K by decapsulating with the KEM secret key. For example, one can view the classical “signed Diffie-Hellman” as a specific instantiation of this paradigm, by considering the Diffie-Hellman protocol as the ElGamal KEM.

Our approach to construct efficient AKE protocols with tight security is based on this KEM + 2 × SIG paradigm. Given a tightly MU-EUF-CMA^{corr} secure signature scheme, it remains to determine suitable security notions for the underlying

¹ Our signatures are only re-randomizable over all strings output by the signing algorithm. The impossibility result from [3] requires uniform re-randomizability over all strings accepted by the verification algorithm, which does not hold for our scheme.

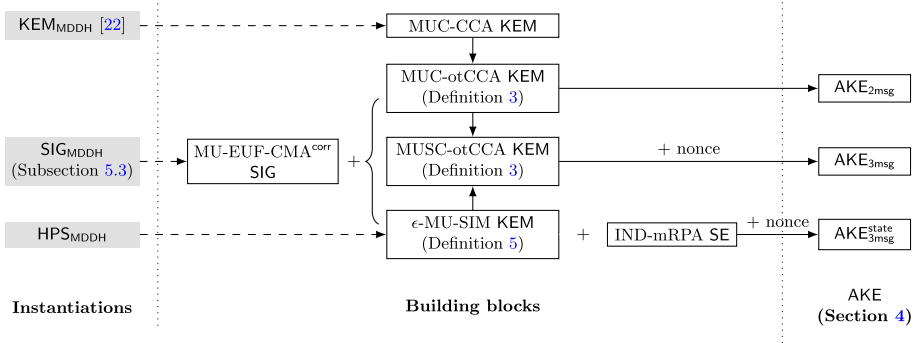


Fig. 2. Schematic overview of our AKE constructions.

KEM, which finds a balance between two properties. The security notion must be *strong enough* to enable a *tight* security proof in presence of adaptive session key reveals and possibly even state reveals. At the same time, it must be *weak enough* to be achievable in the standard model. We now sketch the construction of our three AKE protocols along with the corresponding KEM security notions, see also Fig. 2. In terms of AKE security, we consider a generic and versatile security model which provides strong properties, such as full forward security and key-compromise impersonation (KCI) security. “Partnering” of oracles is defined based on *original key partnering* [28]. The model is defined in pseudocode to avoid ambiguity.

- Our first result is a new tight security proof for the two-message protocol $\text{AKE}_{2\text{msg}}$, which follows the $\text{KEM} + 2 \times \text{SIG}$ paradigm. $\text{AKE}_{2\text{msg}}$ is exactly the LLGW protocol [29] and the main technical difficulty is to adopt the LLGW proof strategy from the “Multi-Bit-Guess” to the standard “Single-Bit-Guess” setting. This requires significant modifications to the proof outline and the underlying KEM security definition. Our new proof relies on Multi-User/Challenge one-time CCA (MUC-otCCA) security for KEMs, allowing the adversary to ask many challenge queries but only one decapsulation query per user. Even though this is a slightly weaker version of the standard Multi-User/Challenge CCA (MUC-CCA) security notion for KEMs (allowing for unbounded decapsulation queries [17]), the most efficient instantiations we could find are the MUC-CCA-secure schemes with tight security from [17, 18, 22].²
- Our second result is a three-message protocol $\text{AKE}_{3\text{msg}}$ resisting replay attacks, which extends the $\text{KEM} + 2 \times \text{SIG}$ protocol $\text{AKE}_{2\text{msg}}$ with an additional nonce sent at the beginning of the protocol (“Nonce + KEM + 2 × SIG”). For our security proof we require the KEM security notion of Multi-User Single-Challenge one-time CCA (MUSC-otCCA) security, allowing the adversary to

² We are aware of the generic constructions of bounded-CCA secure KEMs from CPA-secure KEMs [8], but they do not seem to offer tight security in a multi-challenge setting.

- ask only one challenge and one decapsulation query per user. This notion is considerably weaker than MUC-otCCA security and it is achievable from any universal₂ hash proof system [9]. (For example, based on a standard assumption such as Matrix DDH (MDDH) [15] which yields highly efficient KEMs.)
- Our third result is a three-message protocol $\text{AKE}_{3\text{msg}}^{\text{state}}$, which extends the $\text{Nonce} + \text{KEM} + 2 \times \text{SIG}$ protocol $\text{AKE}_{3\text{msg}}$ by encrypting the state with a symmetric encryption (SE) scheme. $\text{AKE}_{3\text{msg}}^{\text{state}}$ has tight security in a very strong model that even allows the adversary to obtain session states of oracles [7]. The fact that the reduction must be able to respond to adaptive queries for session states by an adversary makes it significantly more difficult to achieve *tight* security. Our key technical contribution is a new “Multi-User SIMulatability” (ϵ -MU-SIM) security notion for KEMs, which we also show to be tightly achievable by universal₂ hash proof systems. We stress that the reduction to the security of the symmetric encryption scheme is the only part of the security proof which is *not* tight. We tolerate this, since compensating a security loss for symmetric encryption incurs significantly less performance penalty than for public key primitives.³

Note that our $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$ use nonce to resist replay attacks and admit KEM security with one challenge per user. This can also be achieved generically by assuming synchronized counters between parties, following the approach of [29]. Consequently, we can also use counter instead of nonce in $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$, and obtain two two-message counter-based AKE protocols which have the same efficiency and security as $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$, respectively.

INSTANTIATIONS. Table 1 gives example instantiations of our protocols from universal₂ hash proof systems from the MDDH assumption and compares them to known protocols. The protocols BHJKL [2] and LLGW [29] only offer tight security in the MBG model which implies security in our standard SBG model with a loss of T , the number of test queries [23]. For more details on our instantiations we refer to Sect. 6. Note that there are other works which study AKE in the standard model (e.g., [16, 24]). However, they do not focus on tightness and have a quadratic security loss.

TECHNICAL APPROACH TO AKE. In the following, we give a brief overview of our technical approach to tight security under our SBG-type security definition and show how our protocols prevent replay attacks and support state reveals.

To obtain an AKE protocol with a tight security reduction in the $\text{KEM} + 2 \times \text{SIG}$ framework, we rely on the tight MU-EUF-CMA^{corr} security of the signature scheme to guarantee authentication and deal with corruptions, and on the tight MUC-CCA security of KEM to deal with session key reveals. To this end, recall

³ For instance, `openssl speed aes` shows that AES-256 is only about 1.5 times slower than AES-128 on a standard laptop computer. Given that the cost of symmetric key operations is already small in comparison to the public key operations, we consider this as negligible.

Table 1. Comparison of standard model AKE protocols with full forward security, where T refers to the number of test queries. Protocols $\text{AKE}_{3\text{msg}}^{\text{state}}$ and $\text{AKE}_{2\text{msg}}$ refer to our protocols given in Fig. 1, instantiated from $\mathcal{D}_k\text{-MDDH}$. The column **Communication** counts the communication complexity of the protocols in terms of the number of group elements, exponents and nonces, where we instantiate all protocols with our new signature scheme from Subsect. 5.3. The column **Security Loss** lists the security loss of the reduction in the “Single-Bit-Guess” (SBG) model, ignoring all symmetric bounds.

Protocol	Communication	#Msg.	Assumption	State Reveal	Security Loss
BHJKL [2]	$11 + 11$ $(2k^2 + 6k + 5) + (6k + 9)$	3	SXDH $\mathcal{D}_{k\text{-MDDH}}$	no	$O(\lambda T)$
LLGW [29]	$9 + 10$ $(k^2 + 7k + 1) + (6k + 4)$	2	SXDH $\mathcal{D}_{k\text{-MDDH}}$	no	$O(\lambda T)$
$\text{AKE}_{3\text{msg}}^{\text{state}}$	$8 + 7$ $(5k + 3) + (5k + 2)$	3	SXDH $\mathcal{D}_{k\text{-MDDH}}$	yes	$O(\lambda)$
$\text{AKE}_{2\text{msg}}$ (= LLGW)	$9 + 10$ $(k^2 + 7k + 1) + (6k + 4)$	2	SXDH $\mathcal{D}_{k\text{-MDDH}}$	no	$O(\lambda)$

that the SBG-style security game for MUC-CCA security allows multiple encapsulation and decapsulation queries per user, but considers only a single challenge bit. At the same time, observe that the reduction algorithm can always use the challenge key (which is either the real encapsulated key or a random key) as the session key of the simulated AKE protocol. In combination, these observations immediately lead to a tight security proof for $\text{AKE}_{2\text{msg}}$. We remark that $\text{AKE}_{2\text{msg}}$ can also be proved secure under an even weaker security notion for KEM, namely MUC-otCCA, which allows only one decapsulation query per user. This assumes that parties choose to “close” a session once this session accepts or rejects. In this way we can guarantee that the adversary has only a single opportunity to submit a ciphertext per \hat{pk} .

To prevent replay attacks we make use of an exchange of nonces resulting in protocol $\text{AKE}_{3\text{msg}}$. As a byproduct of using nonces (in combination with the signature scheme), we can now guarantee that the adversary cannot replay any message anymore. This includes \hat{pk} , and thus we can ensure that the simulator only needs to respond to one encapsulation query per \hat{pk} in the security game. In this way we can further weaken the security requirement that we need from the KEM to MUSC-otCCA.

Now, to support state reveals, we use a symmetric encryption scheme SE that is used to encrypt the ephemeral secret key \hat{sk} of each session, similar to [23]. More concretely, we require that the state is computed as $\text{st} = \text{SE.E}(\hat{s}, \hat{sk})$, where \hat{s} is the secret key of SE that is made part of the long-term secret key. This modification yields protocol $\text{AKE}_{3\text{msg}}^{\text{state}}$. Having introduced such a state, we now also consider a security model that allows the adversary to issue state reveal queries to obtain the state st . But now the reduction to the MUSC-otCCA security of

the KEM cannot work as before, since the reduction algorithm cannot output $\text{SE.E}(s, \hat{s}k)$ to the adversary. A natural way to address this problem is to make use of the security of SE, and make the reduction change the state to an encryption of some dummy random key r , i.e., $\text{st} = \text{SE.E}(s, r)$. However, now the SE reduction algorithm is faced with a critical decision: If the adversary asks a state reveal query, should the reduction output $\text{st} = \text{SE.E}(s, \hat{s}k)$ or $\text{st} = \text{SE.E}(s, r)$? It seems that both choices are problematic. If the reduction responds with the encryption of KEM secret key $\hat{s}k$, then the reduction to the KEM will fail in case the adversary asks a test query. If on the other hand the reduction outputs an encryption of a dummy random key, then the reduction will fail in case the adversary queries the secret key via a corrupt query. To solve this problem, the existing approaches rely on a non-committing symmetric encryption scheme that is proven secure in the random oracle model [23].

To obtain a tight security supporting state reveals in the standard model, we enhance the MUSC-otCCA security of KEM to our new ϵ -MU-SIM-security, so that a symmetric encryption scheme SE with comparatively weak security guarantees suffices. The idea is to rely on a security notion for the symmetric encryption scheme that is as weak as possible while still being able to compensate for this via a stronger KEM. Somewhat surprisingly, our proof shows that when relying on an ϵ -MU-SIM-secure KEM, we only need to require IND-mRPA security (indistinguishability against random plaintext attacks) from SE. Such a symmetric encryption scheme can be easily instantiated using any weakly secure (deterministic) encryption scheme like as AES or even using a weak PRF. Let us now describe ϵ -MU-SIM-secure KEM in slightly more detail. In a nutshell, an ϵ -MU-SIM-secure KEM provides the reduction with access to an additional encapsulation algorithm Encap^* that is keyed with the secret key. We have security requirements as follows:

- Computational indistinguishability between Encap and Encap^* : We require that the reduction can switch to using Encap^* without the adversary noticing even given the secret key $\hat{s}k$ of the KEM. In particular, the resulting indistinguishability notion must tightly reduce to an underlying security assumption.
- Statistical ϵ -uniformity: When using the alternative encapsulation mechanism Encap^* , we require that the encapsulated key in the challenge ciphertext c^* will be indistinguishable from random with *statistical distance* ϵ (even if a decapsulation of some distinct ciphertext $c \neq c^*$ of its choice is given). This is particularly useful when aiming at tight security reductions.
- Since we want to apply ϵ -MU-SIM-secure KEMs in a protocol setting with multiple parties, security must in general hold in a multi-user setting.

Fortunately, such a KEM can be instantiated from universal₂ hash proof systems (HPS). In particular, we show that the ϵ -MU-SIM-security is implied by the hardness of subset membership problems and the universal₂-property of HPS.

Our new ϵ -MU-SIM-secure KEM now allows us to avoid the above mentioned decision when dealing with state reveals and in this way opens a new avenue towards a tight security reduction. To this end, we use a novel strategy in our security proof.

1. We first switch from using Encap to Encap^* . By the security properties of our KEM, the adversary cannot notice this, even given \hat{sk} .
2. Next, we replace the session keys of tested sessions with random keys – one user at a time. We apply a hybrid argument over all users. In the η -th hybrid ($\eta = 1, \dots, \mu$ with μ being the number of users), we replace the test session keys related to the η -th user with random keys. We can show that this is not recognizable by the adversary since the key K^* generated by Encap^* is statistically close to uniform even if the adversary gets to see another key for a ciphertext of its choice. We distinguish the following cases.

Case 1: The adversary corrupts the η -th user. For each session related to this user, the adversary can either reveal the session state or test this session, but not both. If the adversary reveals the state, we do not have to replace the session key at all, so the change is in fact only a conceptual one. If the session is tested, the adversary does not know the state $\text{SE.E}(s, \hat{sk})$ and thus we can replace the session key by exploiting the ϵ -uniformity of Encap^* .

Case 2: The adversary does not corrupt the η -th user. In this case, we rely on the IND-mRPA security of SE and replace \hat{sk} in the encrypted state with a random dummy key for this user. Then, we can use ϵ -uniformity to replace all tested keys for that user with random keys, as the state does not contain any information about \hat{sk} . After that, we have to switch back to using the original state encryption mechanism and encrypt the real secret key \hat{sk} , getting ready for the next hybrid.

After μ hybrids, we change all tested keys to random. At this point the adversary has no advantage in the security game.

Overall, this security proof loses a factor of 2μ – but only when reducing to the IND-mRPA security of the symmetric encryption scheme. All other steps of the proof feature tight security reductions.

2 Security Notions for KEMs

2.1 Preliminaries

Let \emptyset denote an empty string. 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 $x \leftarrow_s \mathcal{X}$ the procedure of sampling x from set \mathcal{X} uniformly at random. If \mathcal{D} is distribution, $x \leftarrow \mathcal{D}$ means that x is sampled according to \mathcal{D} . All our algorithms are probabilistic unless states otherwise. We use $y \leftarrow_s \mathcal{A}(x)$ to define the random variable y obtained by executing algorithm \mathcal{A} on input x . We use $y \in \mathcal{A}(x)$ to indicate that y lies in the support of $\mathcal{A}(x)$. If \mathcal{A} is deterministic we write $y \leftarrow \mathcal{A}(x)$. We also use $y \leftarrow \mathcal{A}(x; r)$ to make the random coins r used in the probabilistic computation explicit. Denote by $\mathbf{T}(\mathcal{A})$ the running time of \mathcal{A} . For two distributions X and Y , the statistical distance between them is defined by $\Delta(X; Y) := \frac{1}{2} \cdot \sum_x |\Pr[X = x] - \Pr[Y = x]|$, and conditioned on $Z = z$, the statistical distance between X and Y is denoted by $\Delta(X; Y|Z = z)$. For $0 \leq \epsilon \leq 1$, X and Y are said to be ϵ -close, denoted by $X \approx_\epsilon Y$, if $\Delta(X; Y) \leq \epsilon$.

Definition 1 (Collision-resistant hash functions). A family of hash functions \mathcal{H} is collision resistant if for any adversary \mathcal{A} ,

$$\text{Adv}_{\mathcal{H}}^{\text{cr}}(\mathcal{A}) := \Pr[x_1 \neq x_2 \wedge H(x_1) = H(x_2) | (x_1, x_2) \leftarrow_{\mathcal{S}} \mathcal{A}(H), H \leftarrow_{\mathcal{S}} \mathcal{H}].$$

2.2 Key Encapsulation Mechanisms

Definition 2 (KEM). A key encapsulation mechanism (KEM) scheme $\text{KEM} = (\text{KEM.Setup}, \text{KEM.Gen}, \text{Encap}, \text{Decap})$ consists of four algorithms:

- **KEM.Setup:** The setup algorithm outputs public parameters pp_{KEM} , which determine an encapsulation key space \mathcal{K} , public key $\mathcal{P}\mathcal{K}$ and secret key spaces $\mathcal{P}\mathcal{K} \times \mathcal{S}\mathcal{K}$, and a ciphertext space $\mathcal{C}\mathcal{T}$.
- **KEM.Gen(pp_{KEM}):** Taking pp_{KEM} as input, the key generation algorithm outputs a pair of public key and secret key $(pk, sk) \in \mathcal{P}\mathcal{K} \times \mathcal{S}\mathcal{K}$. W.l.o.g., we assume that **KEM.Gen** first samples $sk \leftarrow_{\mathcal{S}} \mathcal{S}\mathcal{K}$ uniformly, and then computes pk from sk deterministically via a polynomial-time algorithm **KEM.PK**, i.e., $pk := \text{KEM.PK}(sk)$. This is reasonable since we can always take the randomness used by **KEM.Gen** as the secret key.
- **Encap(pk):** Taking pk as input, the encapsulation algorithm outputs a pair of ciphertext $c \in \mathcal{C}\mathcal{T}$ and encapsulated key $K \in \mathcal{K}$.
- **Decap(sk, c):** Taking as input sk and c , the deterministic decapsulation algorithm outputs $K \in \mathcal{K} \cup \{\perp\}$.

We require that for all $\text{pp}_{\text{KEM}} \in \text{KEM.Setup}$, $(pk, sk) \in \text{KEM.Gen}(\text{pp}_{\text{KEM}})$, $(c, K) \in \text{Encap}(pk)$, it holds that $\text{Decap}(sk, c) = K$.

We define two security notions for KEMs, the first one in the Multi-User/Challenge (MUC) setting, the second one in the Multi-User and Single Challenge (MUSC) setting. Both notions only allow for one single decapsulation query per user.

Definition 3 (MUC-otCCA/MUSC-otCCA Security for KEM). To **KEM**, the number of users $\mu \in \mathbb{N}$, and an adversary \mathcal{A} we associate the advantage functions $\text{Adv}_{\text{KEM}, \mu}^{\text{muc-otcca}}(\mathcal{A}) := \left| \Pr[\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{muc-otcca}} \Rightarrow 1] - \frac{1}{2} \right|$ and $\text{Adv}_{\text{KEM}, \mu}^{\text{musc-otcca}}(\mathcal{A}) := \left| \Pr[\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{musc-otcca}} \Rightarrow 1] - \frac{1}{2} \right|$, where the experiments are defined in Fig. 3.

Below we recall the definition of the diversity property from [29].

Definition 4 (γ -Diversity of KEM). A KEM scheme **KEM** is called γ -diverse if for all $\text{pp}_{\text{KEM}} \in \text{KEM.Setup}$, it holds that

$$\Pr \left[\begin{array}{l} (pk, sk) \leftarrow_{\mathcal{S}} \text{KEM.Gen}(\text{pp}_{\text{KEM}}); \\ r, r' \leftarrow_{\mathcal{R}}; (c, K) \leftarrow \text{Encap}(pk; r); (c', K') \leftarrow \text{Encap}(pk; r') : K = K' \end{array} \right] \leq 2^{-\gamma},$$

$$\Pr \left[\begin{array}{l} (pk, sk) \leftarrow_{\mathcal{S}} \text{KEM.Gen}(\text{pp}_{\text{KEM}}); (pk', sk') \leftarrow_{\mathcal{S}} \text{KEM.Gen}(\text{pp}_{\text{KEM}}); \\ r \leftarrow_{\mathcal{R}}; (c, K) \leftarrow \text{Encap}(pk; r); (c', K') \leftarrow \text{Encap}(pk'; r) : K = K' \end{array} \right] \leq 2^{-\gamma},$$

where \mathcal{R} is the randomness space of **Encap**. If $\gamma = \log |\mathcal{K}|$, then **KEM** is perfectly diverse.

$\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{muc-otcca}}$: $\text{pp}_{\text{KEM}} \leftarrow_{\$} \text{KEM.Setup}$ For $i \in [\mu]$: $(pk_i, sk_i) \leftarrow_{\$} \text{KEM.Gen}(\text{pp}_{\text{KEM}})$ $\text{EncList} := \emptyset$ //Records the encapsulation queries $b \leftarrow_{\$} \{0, 1\}$ //Single challenge bit $\text{PKList} := \{pk_i\}_{i \in [\mu]}$ $b' \leftarrow_{\$} \mathcal{A}^{\mathcal{O}_{\text{ENCAP}}(\cdot), \mathcal{O}_{\text{DECAP}}(\cdot, \cdot)}(\text{pp}_{\text{KEM}}, \text{PKList})$ If $b' = b$: Return 1; Else: Return 0	$\mathcal{O}_{\text{ENCAP}}^b(i)$: //at most once per user i $(c, K) \leftarrow_{\$} \text{Encap}(pk_i)$ $\text{EncList} := \text{EncList} \cup \{(i, c)\}$ $K_0 := K; K_1 \leftarrow_{\$} \mathcal{K}$ Return (c, K_b) <hr/> $\mathcal{O}_{\text{DECAP}}(i, c')$: // at most once per user i If $(i, c') \notin \text{EncList}$: Return $K' \leftarrow_{\$} \text{Decap}(sk_i, c')$ Else: Return \perp
--	--

Fig. 3. The MUC-otCCA security experiment $\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{muc-otcca}}$ and the MUSC-otCCA security experiment $\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{musc-otcca}}$ of KEM, where in the latter the adversary can query the encapsulation oracle only once for each user.

We also propose a new security notion for KEMs called ϵ -MU-SIM (ϵ -multi-user simulatable) security. Jumping ahead, ϵ -MU-SIM secure KEMs will serve as the main building block in our generic AKE construction with state reveal later. We present the formal definition of ϵ -MU-SIM security (Definition 5). We also present simple constructions of ϵ -MU-SIM secure KEMs from universal₂-HPS in the full version [21].

Informally, ϵ -MU-SIM security requires that there exists a simulated encapsulation algorithm $\text{Encap}^*(sk)$ which returns simulated ciphertext/key pairs (c^*, K^*) satisfying the following two properties. Firstly, they should be computationally indistinguishable from real ciphertext/key pairs. Secondly, given c^* and an arbitrary single decryption query, the simulated key K^* should be ϵ -close to uniform.

Definition 5 (ϵ -MU-SIM Security for KEM). *We require that there exists a simulated encapsulation algorithm $\text{Encap}^*(sk)$ which takes the secret key sk as input, and outputs a pair of simulated $c^* \in \mathcal{CT}$ and simulated $K^* \in \mathcal{K}$. For ϵ -uniformity we require that for any (unbounded) adversary \mathcal{A} , it holds that*

$$\left| \begin{aligned} & \Pr[c \leftarrow_{\$} \mathcal{A}(pk, c^*, K^*) : c \neq c^* \wedge \mathcal{A}(pk, c^*, K^*, \text{Decap}(sk, c)) \Rightarrow 1] \\ & - \Pr[c \leftarrow_{\$} \mathcal{A}(pk, c^*, R) : c \neq c^* \wedge \mathcal{A}(pk, c^*, R, \text{Decap}(sk, c)) \Rightarrow 1] \end{aligned} \right| \leq \epsilon, \quad (1)$$

where the probability is over $\text{pp}_{\text{KEM}} \leftarrow_{\$} \text{KEM.Setup}$, $(pk, sk) \leftarrow_{\$} \text{KEM.Gen}(\text{pp}_{\text{KEM}})$, $(c^*, K^*) \leftarrow_{\$} \text{Encap}^*(sk)$, $R \leftarrow_{\$} \mathcal{K}$ and the internal randomness of \mathcal{A} .

Furthermore, to KEM, a simulated encapsulation algorithm Encap^* , an adversary \mathcal{A} , and $\mu \in \mathbb{N}$ we associate the advantage function $\text{Adv}_{\text{KEM}, \text{Encap}^*, \mu}^{\text{mu-sim}}(\mathcal{A}) :=$

$$\left| \Pr \left[\mathcal{A}(\{pk_i, sk_i, c_i^{(0)}, K_i^{(0)}\}_{i \in [\mu]}) \Rightarrow 1 \right] - \Pr \left[\mathcal{A}(\{pk_i, sk_i, c_i^{(1)}, K_i^{(1)}\}_{i \in [\mu]}) \Rightarrow 1 \right] \right|, \quad (2)$$

where $\text{pp}_{\text{KEM}} \leftarrow_{\$} \text{KEM.Setup}$, $(pk_i, sk_i) \leftarrow_{\$} \text{KEM.Gen}(\text{pp}_{\text{KEM}})$, $(c_i^{(0)}, K_i^{(0)}) \leftarrow_{\$} \text{Encap}(pk_i)$, and $(c_i^{(1)}, K_i^{(1)}) \leftarrow_{\$} \text{Encap}^*(sk_i)$ for $\forall i \in [\mu]$.

Note that ϵ -MU-SIM security tightly implies MUSC-otCCA^{rev&corr} security which is a stronger variant of MUSC-otCCA security supporting key reveal and user

corrupt queries. Reveal and corrupt queries can be tolerated since in the security experiment (2), adversary \mathcal{A} also obtains secret keys sk_1, \dots, sk_μ . By (1) one can see that one single decapsulation query is supported. In particular, ϵ -MU-SIM security tightly implies MUSC-otCCA security. In the full version [21], we will define universal₂ hash proof systems, construct HPS_{MDDH} schemes from the MDDH assumptions, and show how they imply ϵ -MU-SIM secure KEMs.

3 Authenticated Key Exchange

3.1 Definition of Authenticated Key Exchange

Definition 6 (AKE). *An authenticated key exchange (AKE) scheme $\text{AKE} = (\text{AKE.Setup}, \text{AKE.Gen}, \text{AKE.Protocol})$ consists of two probabilistic algorithms and an interactive protocol.*

- AKE.Setup : *The setup algorithm outputs the public parameter pp_{AKE} .*
- $\text{AKE.Gen}(\text{pp}_{\text{AKE}}, P_i)$: *The generation algorithm takes as input pp_{AKE} and a party P_i , and outputs a key pair (pk_i, sk_i) .*
- $\text{AKE.Protocol}(P_i(\text{res}_i) \rightleftharpoons P_j(\text{res}_j))$: *The protocol involves two parties P_i and P_j , who have access to their own resources, $\text{res}_i := (sk_i, \text{pp}_{\text{AKE}}, \{pk_u\}_{u \in [\mu]})$ and $\text{res}_j := (sk_j, \text{pp}_{\text{AKE}}, \{pk_u\}_{u \in [\mu]})$, respectively. Here μ is the total number of users. After execution, P_i outputs a flag $\Psi_i \in \{\emptyset, \text{accept}, \text{reject}\}$, and a session key k_i (k_i might be the empty string \emptyset), and P_j outputs (Ψ_j, k_j) similarly.*

Correctness of AKE. For any distinct and honest parties P_i and P_j , they share the same session key after the execution of $\text{AKE.Protocol}(P_i(\text{res}_i) \rightleftharpoons P_j(\text{res}_j))$, i.e., $\Psi_i = \Psi_j = \text{accept}, k_i = k_j \neq \emptyset$.

3.2 Security Model of AKE

We will adapt the security model formalized by [2, 19, 28], which in turn followed the model proposed by Bellare and Rogaway [5]. We also include replay attacks [29] and multiple test queries with respect to the same random bit [23].

First, we will define oracles and their static variables in the model. Then we describe the security experiment and the corresponding security notions.

Oracles. Suppose there are at most μ users P_1, P_2, \dots, P_μ , and each user will involve at most ℓ instances. P_i is formalized by a series of oracles, $\pi_i^1, \pi_i^2, \dots, \pi_i^\ell$. Oracle π_i^s formalizes P_i 's execution of the s -th protocol instance.

Each oracle π_i^s has access to P_i 's resource $\text{res}_i := (sk_i, \text{pp}_{\text{AKE}}, \text{PKList} := \{pk_u\}_{u \in [\mu]})$. π_i^s also has its own variables $\text{var}_i^s := (\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)$.

- st_i^s : State information that has to be stored between two rounds in order to execute the protocol.
- Pid_i^s : The intended communication peer's identity.

- $k_i^s \in \mathcal{K}$: The session key computed by π_i^s . Here \mathcal{K} is the session key space. We assume that $\emptyset \in \mathcal{K}$.
- $\Psi_i^s \in \{\emptyset, \mathbf{accept}, \mathbf{reject}\}$: Ψ_i^s indicates whether π_i^s has completed the protocol execution and accepted k_i^s .

At the beginning, $(st_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)$ are initialized to $(\emptyset, \emptyset, \emptyset, \emptyset)$. We declare that $k_i^s \neq \emptyset$ if and only if $\Psi_i^s = \mathbf{accept}$.

Security Experiment. To define the security notion of AKE, we first formalize the security experiment $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$ with the help of the oracles defined above. $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$ is a game played between an AKE challenger \mathcal{C} and an adversary \mathcal{A} . \mathcal{C} will simulate the executions of the ℓ protocol instances for each of the μ users with oracles π_i^s . We give a formal description in Fig. 4.

Adversary \mathcal{A} may copy, delay, erase, replay, and interpolate the messages transmitted in the network. This is formalized by the query **Send** to oracle π_i^s . With **Send**, \mathcal{A} can send arbitrary messages to any oracle π_i^s . Then π_i^s will execute the AKE protocol according to the protocol specification for P_i . The **StateReveal**(i, s) oracle allows \mathcal{A} to reveal π_i^s 's session state.

We also allow the adversary to observe session keys of its choices. This is reflected by a **SessionKeyReveal** query to oracle π_i^s .

A **Corrupt** query allows \mathcal{A} to corrupt a party P_i and get its long-term secret key sk_i . With a **RegisterCorrupt** query, \mathcal{A} can register a new party without public key certification. The public key is then known to all other users.

We introduce a **Test** query to formalize the pseudorandomness of k_i^s . Therefore, the challenger chooses a bit $b \leftarrow_s \{0, 1\}$ at the beginning of the experiment. When \mathcal{A} issues a **Test** query for π_i^s , the oracle will return \perp if the session key k_i^s is not generated yet. Otherwise, π_i^s will return k_i^s or a truly random key, depending on b . The task of \mathcal{A} is to tell whether the key is the true session key or a random key. The adversary is allowed to make multiple test queries.

Formally, the queries by \mathcal{A} are described as follows.

- **Send**(i, s, j, msg): If $\text{msg} = \top$, it means that \mathcal{A} asks oracle π_i^s to send the first protocol message to P_j . Otherwise, \mathcal{A} impersonates P_j to send message msg to π_i^s . Then π_i^s executes the AKE protocol with msg as P_i does, computes a message msg' , and updates its own variables $\text{var}_i^s = (st_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)$. The output message msg' is returned to \mathcal{A} .
If **Send**(i, s, j, msg) is the τ -th query asked by \mathcal{A} and π_i^s changes Ψ_i^s to **accept** after that, then we say that π_i^s is τ -*accepted*.
- **Corrupt**(i): \mathcal{C} reveals party P_i 's long-term secret key sk_i to \mathcal{A} . After corruption, $\pi_i^1, \dots, \pi_i^\ell$ will stop answering any query from \mathcal{A} .
If **Corrupt**(i) is the τ -th query asked by \mathcal{A} , we say that P_i is τ -*corrupted*.
If \mathcal{A} has never asked **Corrupt**(i), we say that P_i is ∞ -*corrupted*.
- **RegisterCorrupt**(i, pk_i): It means that \mathcal{A} registers a new party P_i ($i > \mu$). \mathcal{C} distributes (P_i, pk_i) to all users. In this case, we say that P_i is 0-*corrupted*.
- **StateReveal**(i, s): The query means that \mathcal{A} asks \mathcal{C} to reveal π_i^s 's session state. \mathcal{C} returns st_i^s to \mathcal{A} .

<pre> EXP_{AKE,μ,ℓ,ℓ,ℓ,ℓ,ℓ}: pp_{AKE} ← AKE.Setup For i ∈ [μ]: (pk_i, sk_i) ← AKE.Gen(pp_{AKE}, P_i); crp_i := false //Corruption variable PKList := {pk_i}_{i∈[μ]}; b ←_s {0, 1} For (i, s) ∈ [μ] × [ℓ]: var_i^s := (st_i^s, Pid_i^s, k_i^s, Ψ_i^s) := (0, 0, 0, 0); Aflag_i^s := false //Whether Pid_i^s is corrupted when π_i^s accepts T_i^s := false; kRev_i^s := false // Test, Key Reveal variables stRev_i^s := false, FirstAcc_i^s := 0 // State Reveal & First Acceptance variables b* ← $\mathcal{A}^{\mathcal{O}_{AKE}(\cdot)}$(pp_{AKE}, PKList) Win_{Auth} := false Win_{Auth} := true, If ∃(i, s) ∈ [μ] × [ℓ] s.t. (1) Ψ_i^s = accept //π_i^s is τ-accepted (2) Aflag_i^s = false //P_j is τ-corrupted with j := Pid_i^s and τ̂ > τ (3) (3.1) ∨ (3.2) ∨ (3.3), Let j := Pid_i^s (3.1) ∃ t ∈ [ℓ] s.t. Partner(π_i^s ← π_j^t) (3.2) ∃ t ∈ [ℓ], (j', t') ∈ [μ] × [ℓ] with (j, t) ≠ (j', t') s.t. Partner(π_i^s ← π_j^t) ∧ Partner(π_i^s ← π_{j'}^{t'}) (3.3) ∃ t ∈ [ℓ], (i', s') ∈ [μ] × [ℓ] with (i, s) ≠ (i', s') s.t. Partner(π_i^s ← π_j^t) ∧ Partner(π_{i'}^{s'} ← π_j^t) //Replay attacks Win_{Ind} := false If b* = b: Win_{Ind} := true; Return 1 Else: Return 0 Partner(π_i^s ← π_j^t): //Checking whether Partner(π_i^s ← π_j^t) If π_i^s sent the first message and k_i^s = K(π_i^s, π_j^t) ≠ ∅: Return 1 If π_j^t received the first message and k_j^t = K(π_j^t, π_i^s) ≠ ∅: Return 1 Return 0 π_i^s(msg, j): //π_i^s executes AKE according to the protocol specification If Pid_i^s = ∅: Pid_i^s := j If Pid_i^s = j: π_i^s receives msg and uses res_i, var_i^s to generate the next message msg' of AKE, and updates (st_i^s, Pid_i^s, k_i^s, Ψ_i^s); If msg = ⊤: π_i^s generates the first message msg' as initiator; If msg is the last message of AKE: msg' := ∅; Return msg' If Pid_i^s ≠ j: Return ⊥ O_{AKE}(query): If query=RegisterCorrupt(u, pk_u): If u ∈ [μ]: Return ⊥ PKList := PKList ∪ {pk_u} crp_u := true Return PKList </pre>	<pre> O_{AKE}(query): If query=Send(i, s, j, msg): If Ψ_i^s = accept: Return ⊥ msg' ← π_i^s(msg, j) If Ψ_i^s = accept: If crp_j = true: Aflag_i^s := true; // Determine whether π_i^s accepts before its partner If crp_j = false ∧ ∃t ∈ [ℓ] s.t. Partner(π_i^s ← π_j^t): If Ψ_j^t ≠ accept: FirstAcc_i^s := true; FirstAcc_j^t := false If Ψ_j^t = accept: FirstAcc_i^s := false; FirstAcc_j^t := true Return msg' If query=Corrupt(i): If i ∉ [μ]: Return ⊥ For s ∈ [ℓ] If FirstAcc_i^s = false ∧ stRev_i^s = true: If T_i^s = true: Return ⊥; //avoid TA6 If ∃t ∈ [ℓ] s.t. Partner(π_i^s ← π_j^t): If T_j^t = true: Return ⊥; //avoid TA7 crp_i := true Return sk_i If query=SessionKeyReveal(i, s): If Ψ_i^s ≠ accept: Return ⊥ If T_i^s = true: Return ⊥ //avoid TA2 Let j := Pid_i^s If ∃t ∈ [ℓ] s.t. Partner(π_i^s ↔ π_j^t): If T_j^t = true: Return ⊥ //avoid TA4 kRev_i^s := true; Return k_i^s If query=StateReveal(i, s) If FirstAcc_i^s = false ∧ crp_i = true: If T_i^s = true: Return ⊥; //avoid TA6 Let j := Pid_i^s If ∃t ∈ [ℓ] s.t. Partner(π_i^s ← π_j^t): If T_j^t = true: Return ⊥; //avoid TA7 stRev_i^s := true; Return st_i^s If query=Test(i, s): If Ψ_i^s ≠ accept ∨ Aflag_i^s = true ∨ kRev_i^s = true ∨ T_i^s = true: Return ⊥ //avoid TA1, TA2, TA3 If FirstAcc_i^s = false: If crp_i = true ∧ stRev_i^s = true: Return ⊥ //avoid TA6 Let j := Pid_i^s If ∃t ∈ [ℓ] s.t. Partner(π_i^s ↔ π_j^t): If kRev_j^t = true ∨ T_j^t = true: Return ⊥ //avoid TA4, TA5 If ∃t ∈ [ℓ] s.t. Partner(π_i^s ← π_j^t): If FirstAcc_j^t = false ∧ crp_j = true ∧ stRev_j^t = true: Return ⊥ //avoid TA7 T_i^s := true; k₀ := k_i^s; k₁ ←_s K; Return k₀ </pre>
--	---

Fig. 4. The security experiments $\text{Exp}_{\text{AKE},\mu,\ell,\ell,\ell,\ell,\ell}$, $\text{Exp}_{\text{AKE},\mu,\ell,\ell,\ell,\ell,\ell}^{\text{replay}}$ (both without red text) and $\text{Exp}_{\text{AKE},\mu,\ell,\ell,\ell,\ell,\ell}^{\text{replay,state}}$ (with red text). The list of trivial attacks is given in Table 2.

- **SessionKeyReveal**(i, s): The query means that \mathcal{A} asks \mathcal{C} to reveal π_i^s 's session key. If $\Psi_i^s \neq \mathbf{accept}$, \mathcal{C} returns \perp . Otherwise, \mathcal{C} returns the session key k_i^s of π_i^s .
- **Test**(i, s): If $\Psi_i^s \neq \mathbf{accept}$, \mathcal{C} returns \perp . Otherwise, \mathcal{C} sets $k_0 = k_i^s$, samples $k_1 \leftarrow_s \mathcal{K}$, and returns k_b to \mathcal{A} . We require that \mathcal{A} can ask **Test**(i, s) to each oracle π_i^s only once.

Informally, the pseudorandomness of k_i^s asks that any PPT adversary \mathcal{A} with access to **Test**(i, s) cannot distinguish k_i^s from a uniformly random key. Yet, we have to exclude some trivial attacks. We will define them later and first introduce partnering.

Definition 7 (Original Key [28]). For two oracles π_i^s and π_j^t , the original key, denoted as $K(\pi_i^s, \pi_j^t)$, is the session key computed by the two peers of the protocol under a passive adversary only, where π_i^s is the initiator.

Remark 1. We note that $K(\pi_i^s, \pi_j^t)$ is determined by the identities of P_i and P_j and the internal randomness.

Definition 8 (Partner [28]). Let $K(\cdot, \cdot)$ denote the original key function. We say that an oracle π_i^s is partnered to π_j^t , denoted as $\mathbf{Partner}(\pi_i^s \leftarrow \pi_j^t)$ ⁴, if one of the following requirements holds:

- π_i^s has sent the first message and $k_i^s = K(\pi_i^s, \pi_j^t) \neq \emptyset$, or
- π_i^s has received the first message and $k_i^s = K(\pi_j^t, \pi_i^s) \neq \emptyset$.

We write $\mathbf{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$ if $\mathbf{Partner}(\pi_i^s \leftarrow \pi_j^t)$ and $\mathbf{Partner}(\pi_j^t \leftarrow \pi_i^s)$.

Trivial Attacks. In order to prevent the adversary from trivial attacks, we keep track of the following variables for each party P_i and oracle π_i^s :

- crp_i : whether P_i is corrupted.
- \mathbf{Aflag}_i^s : whether the intended partner is corrupted when π_i^s accepts.
- T_i^s : whether π_i^s was tested.
- $kRev_i^s$: whether the session key k_i^s was revealed.
- $stRev_i^s$: whether the session state st_i^s was revealed.
- $\mathbf{FirstAcc}_i^s$: whether P_i or its partner is the first to accept the key in the session.

Based on that we give a list of trivial attacks **TA1-TA7** in Table 2.

Remark 2. We introduced variable $\mathbf{FirstAcc}$ to indicate whether the party or its partner is the first to accept the key. This is used to determine whether the state of an oracle is allowed to be revealed when the oracle or its partner is tested.

⁴ The arrow notion $\pi_i^s \leftarrow \pi_j^t$ means π_i^s (not necessarily π_j^t) has computed and accepted the original key.

- In general, the session key of the party which accepts the key after its partner (i.e., $FirstAcc = \mathbf{false}$), by the correctness of AKE, must be identical to its partner's. Thus, the session key is fully determined by the state and long-term key of that party (as well as transcripts).
- However, the session key of the party which accepts the key before its partner (i.e., $FirstAcc = \mathbf{true}$) might involve fresh randomness beyond its state and long-term key.

Thus, it is a trivial attack to reveal the state and the long-term key of the same oracle, if the oracle or its partner is tested and the oracle accepts the key after its partner (i.e., $FirstAcc = \mathbf{false}$). This is a minimal trivial attack regarding state reveal⁵, and it is formalized as **TA6** and **TA7** in Table 2.

The following definition also captures replay attacks. Given $\text{Partner}(\pi_{i'}^s \leftarrow \pi_j^t)$, a successful replay attack means that \mathcal{A} resends to π_i^s the messages, which were sent from π_j^t to $\pi_{i'}^s$, and π_i^s is fooled to compute a session key, i.e., $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$. Note that a stateless 2-pass AKE protocol cannot be secure against replay attacks [29]. Therefore, we also define security without replay attacks in Definition 11.

Furthermore, we distinguish between security with state reveals (Definition 9) and without state reveals (Definition 10), where in the latter the adversary cannot query the session state of an oracle.

Table 2. Trivial attacks **TA1-TA7** for security experiments $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$, $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay}}$ and $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}}$, where **TA6** and **TA7** are only defined in $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}}$. Note that “ $\text{Aflag}_i^s = \mathbf{false}$ ” is implicitly contained in **TA2-TA7** because of **TA1**.

Types	Trivial attacks	Explanation
TA1	$T_i^s = \mathbf{true} \wedge \text{Aflag}_i^s = \mathbf{true}$	π_i^s is tested but π_i^s 's partner is corrupted when π_i^s accepts session key k_i^s
TA2	$T_i^s = \mathbf{true} \wedge kRev_i^s = \mathbf{true}$	π_i^s is tested and its session key k_i^s is revealed
TA3	$T_i^s = \mathbf{true}$ when $\text{Test}(i, s)$ is queried	$\text{Test}(i, s)$ is queried at least twice
TA4	$T_i^s = \mathbf{true} \wedge \text{Partner}(\pi_i^s \leftrightarrow \pi_j^t) \wedge kRev_j^t = \mathbf{true}$	π_i^s is tested, π_i^s and π_j^t are partnered to each other, and π_j^t 's session key k_j^t is revealed
TA5	$T_i^s = \mathbf{true} \wedge \text{Partner}(\pi_i^s \leftrightarrow \pi_j^t) \wedge T_j^t = \mathbf{true}$	π_i^s is tested, π_i^s and π_j^t are partnered to each other, and π_j^t is tested
TA6	$T_i^s = \mathbf{true} \wedge FirstAcc_i^s = \mathbf{false}$ $\wedge stRev_i^s = \mathbf{true} \wedge crp_i = \mathbf{true}$	π_i^s is tested, π_i^s accepts its key after its partner, and π_i^s is both corrupted and has its state st_i^s revealed
TA7	$T_i^s = \mathbf{true} \wedge \text{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge FirstAcc_j^t = \mathbf{false}$ $\wedge stRev_j^t = \mathbf{true} \wedge crp_j = \mathbf{true}$	π_i^s is tested, π_i^s accepts its session key before its partner, but its partner π_j^t is both corrupted and state revealed

Definition 9 (Security of AKE with Replay Attacks and State Reveal).

Let μ be the number of users and ℓ the maximum number of protocol executions

⁵ It is also possible to define the trivial attack regardless of $FirstAcc$, but our definition of **TA6** and **TA7** is minimal and makes our security model stronger.

per user. The security experiment $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay,state}}$ (see Fig. 4) is played between the challenger \mathcal{C} and the adversary \mathcal{A} .

1. \mathcal{C} runs AKE.Setup to get AKE public parameter pp_{AKE} .
2. For each party P_i , \mathcal{C} runs $\text{AKE.Gen}(\text{pp}_{\text{AKE}}, P_i)$ to get the long-term key pair (pk_i, sk_i) . Next it chooses a random bit $b \leftarrow \{0, 1\}$ and provides \mathcal{A} with the public parameter pp_{AKE} and the list of public keys $\text{PKList} := \{pk_i\}_{i \in [\mu]}$.
3. \mathcal{A} asks \mathcal{C} Send , Corrupt , RegisterCorrupt , SessionKeyReveal , StateReveal and Test queries adaptively.
4. At the end of the experiment, \mathcal{A} terminates with an output b^* .

- **Strong Authentication.** Let Win_{Auth} denote the event that \mathcal{A} breaks authentication in the security experiment. Win_{Auth} happens iff $\exists (i, s) \in [\mu] \times [\ell]$ s.t.
 - (1) π_i^s is τ -accepted.
 - (2) P_j is $\hat{\tau}$ -corrupted with $j := \text{Pid}_i^s$ and $\hat{\tau} > \tau$.
 - (3) Either (3.1) or (3.2) or (3.3) happens⁶. Let $j := \text{Pid}_i^s$.
 - (3.1) There is no oracle π_j^t that π_i^s is partnered to.
 - (3.2) There exist two distinct oracles π_j^t and $\pi_j^{t'}$, to which π_i^s is partnered.
 - (3.3) There exist two oracles $\pi_{i'}^{s'}$ and π_j^t with $(i', s') \neq (i, s)$, such that both π_i^s and $\pi_{i'}^{s'}$ are partnered to π_j^t .
- **Indistinguishability.** Let Win_{Ind} denote the event that \mathcal{A} breaks indistinguishability in $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay,state}}$ above. Let b^* be \mathcal{A} 's output. Then Win_{Ind} happens iff $b^* = b$. Trivial attacks are already considered during the execution of the experiment. A list of trivial attacks is given in Table 2.

Note that $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay,state}} \Rightarrow 1$ iff Win_{Ind} happens. Hence, the advantage of \mathcal{A} is defined as

$$\begin{aligned} \text{Adv}_{\text{AKE},\mu,\ell}^{\text{replay,state}}(\mathcal{A}) &:= \max\{\Pr[\text{Win}_{\text{Auth}}], |\Pr[\text{Win}_{\text{Ind}}] - 1/2|\} \\ &= \max\{\Pr[\text{Win}_{\text{Auth}}], |\Pr[\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay,state}} \Rightarrow 1] - 1/2|\}. \end{aligned}$$

Definition 10 (Security of AKE with Replay Attacks and without State Reveal). The security experiment $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay}}$ (see Fig. 4) is defined like $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay,state}}$ except that we disallow state reveal queries. Similarly, the advantage of an adversary \mathcal{A} in $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay}}$ is defined as

$$\text{Adv}_{\text{AKE},\mu,\ell}^{\text{replay}}(\mathcal{A}) := \max\{\Pr[\text{Win}_{\text{Auth}}], |\Pr[\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay}} \Rightarrow 1] - 1/2|\}.$$

Definition 11 (Security of AKE without Replay Attack and State Reveal). The security experiment $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}$ (see Fig. 4) is defined like $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay,state}}$ except that we disallow replay attacks and state reveal queries. Similarly, the advantage of an adversary \mathcal{A} in $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}$ is defined as

$$\text{Adv}_{\text{AKE},\mu,\ell}(\mathcal{A}) := \max\{\Pr[\text{Win}_{\text{Auth}}], |\Pr[\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1] - 1/2|\}.$$

⁶ Given (1) \wedge (2), (3.1) indicates a successful impersonation of P_j , (3.2) suggests one instance of P_i has multiple partners, and (3.3) corresponds to a successful replay attack.

Remark 3 (Perfect Forward Security and KCI Resistance). The security model of AKE supports (perfect) forward security (a.k.a. forward secrecy [20]). That is, if P_i or its partner P_j has been corrupted at some moment, then the exchanged session keys computed before the corruption remain hidden from the adversary. Meanwhile, π_i^s may be corrupted before $\text{Test}(i, s)$, which provides resistance to key-compromise impersonation (KCI) attacks [25].

4 AKE Protocols

We construct AKE protocols $\text{AKE}_{2\text{msg}}$, $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$ from a signature scheme SIG and a key encapsulation mechanism KEM. Additionally, we use a symmetric encryption scheme SE with key space \mathcal{K}_{SE} to encrypt the state in protocol $\text{AKE}_{3\text{msg}}^{\text{state}}$. Apart from that, $\text{AKE}_{3\text{msg}}^{\text{state}}$ and $\text{AKE}_{3\text{msg}}$ are the same. The protocols are given in Fig. 5.

The setup algorithm generates the public parameter $\text{pp}_{\text{AKE}} := (\text{pp}_{\text{SIG}}, \text{pp}_{\text{KEM}})$ by running SIG.Setup and KEM.Setup . The key generation algorithm inputs the public parameter and a party P_i and generates a signature key pair (vk_i, ssk_i) . In $\text{AKE}_{3\text{msg}}^{\text{state}}$, it also chooses a symmetric key s_i uniformly from the key space \mathcal{K}_{SE} . It returns the public key vk_i and the secret key (ssk_i, s_i) .

The protocol is executed between two parties P_i and P_j . Each party has access to their own resources res_i and res_j which contain the corresponding secret key, the public parameter and a list PKList consisting of the public keys of all parties. Each party initializes its local variables Ψ_i , k_i and st_i with the empty string. To initiate a session in $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$, the party P_j chooses a bitstring N uniformly from $\{0, 1\}^\lambda$ and sends it to P_i . The next message and the first message in protocol $\text{AKE}_{2\text{msg}}$ is sent by P_i . It generates an ephemeral key pair $(\hat{p}k, \hat{s}k)$ by running $\text{KEM.Gen}(\text{pp}_{\text{KEM}})$ and computes a signature σ_1 over the identities of P_i and P_j , the ephemeral public key and the nonce (only in $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$). When using state encryption, it also encrypts the ephemeral secret key using its symmetric key s_i and stores the ciphertext in st_i . It then sends $(\hat{p}k, \sigma_1)$ to P_j . P_j verifies the signature using vk_i and rejects if it is not valid. Otherwise, it continues the protocol by computing $(c, K) \leftarrow_s \text{Encap}(\hat{p}k)$. It computes a signature σ_2 over the identities as well as the previous message, c and the nonce (only in $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$). P_j accepts the session key and sets k_j to K . It sends (c, σ_2) to P_i . P_i verifies the signature and rejects if it is invalid. Otherwise, it retrieves the ephemeral secret key by decrypting the state, computes the session key K from c and accepts.

Correctness. Correctness of $\text{AKE}_{2\text{msg}}$, $\text{AKE}_{3\text{msg}}$ and $\text{AKE}_{3\text{msg}}^{\text{state}}$ follows directly from the correctness of SIG, KEM and SE.

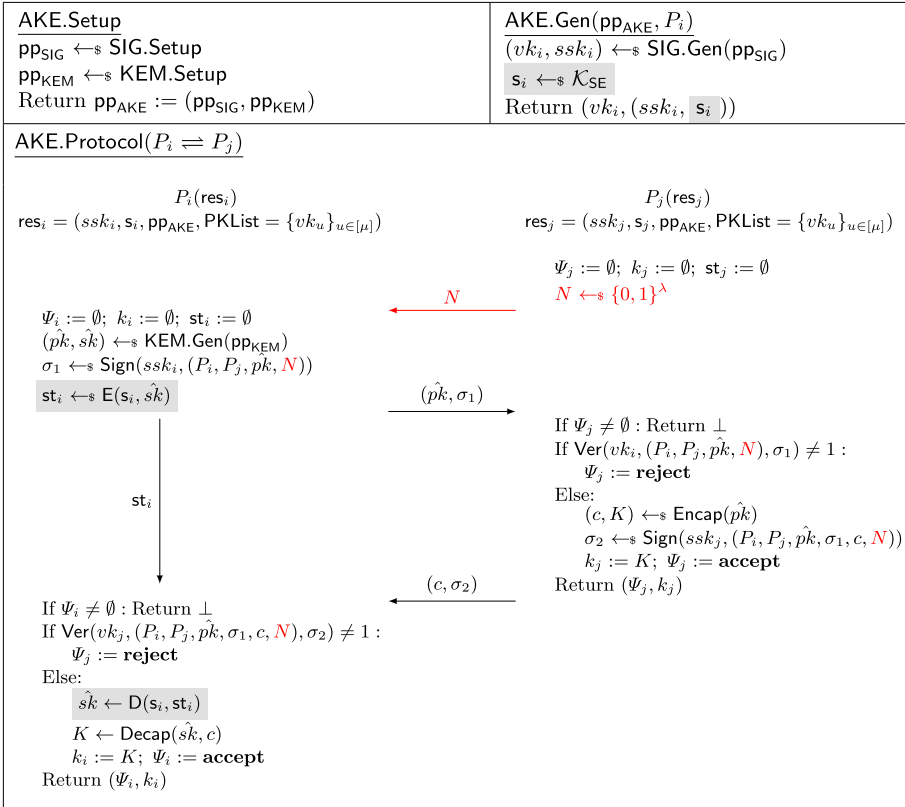


Fig. 5. Generic construction of $\text{AKE}_{2\text{msg}}$ (without red and gray parts), $\text{AKE}_{3\text{msg}}$ (with red and without gray parts) and $\text{AKE}_{3\text{msg}}^{\text{state}}$ (with red and gray parts) from KEM, SIG and SE. Note that the state of P_j only consists of public parts and is therefore omitted here.

Theorem 1 (Security of $\text{AKE}_{3\text{msg}}^{\text{state}}$ with Replay Attacks and State Reveals). For any adversary \mathcal{A} against $\text{AKE}_{3\text{msg}}^{\text{state}}$ with replay attacks and state reveals, there exist an $\text{MU-EUF-CMA}^{\text{corr}}$ adversary \mathcal{B}_{SIG} against SIG, an ϵ - MU-SIM adversary \mathcal{B}_{KEM} against KEM and an IND-mRPA adversary \mathcal{B}_{SE} against SE such that

$$\text{Adv}_{\text{AKE}_{3\text{msg}}^{\text{state}}, \mu, \ell}^{\text{replay, state}}(\mathcal{A}) \leq \text{Adv}_{\text{KEM, Encap}^*, \mu, \ell}^{\text{mu-sim}}(\mathcal{B}_{\text{KEM}}) + 2 \cdot \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + 2\mu \cdot \text{Adv}_{\text{SE}, \mu}^{\text{mrpa}}(\mathcal{B}_{\text{SE}}) + 2\mu\ell \cdot \epsilon + 2(\mu\ell)^2 \cdot 2^{-\gamma} + \mu\ell^2 \cdot 2^{-\lambda},$$

where γ is the diversity parameter of KEM and λ is the length of the nonce N in bits. Furthermore, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{KEM}})$, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{SIG}})$ and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{SE}})$.

We will give a proof sketch below. The formal proof is given in the full version [21].

Proof Sketch. The signatures in the protocol ensure that the adversary can only forward messages for those sessions that it wants to test. Thus the experiment can control all ephemeral public keys \hat{pk} and ciphertexts c that are used for test queries. Due to the nonce, the adversary can also not replay a message containing a particular \hat{pk} . Thus, each \hat{pk} is used in at most one test query.

A party will close a session when it accepts or rejects the session. Thus, the adversary can submit at most one ciphertext c' which is different from the ciphertext used in the test query. Using a session key reveal query, the adversary will only see at most one more key decapsulated with \hat{sk} .

To deal with state reveals, the adversary \mathcal{A} can additionally obtain the state which is the encrypted \hat{sk} . The reduction must know \hat{sk} in order to answer those queries. The simulatability property of KEM ensures that Encap and Encap^* are indistinguishable, even given \hat{sk} . So, we first switch from Encap to Encap^* . Now, we want to replace the session keys of tested sessions with random keys. Therefore, we have to do a hybrid argument over all users. In the η -th hybrid, we replace the test session keys for party P_η . We can show that this is unnoticeable using that the key K^* generated by Encap^* is statistically close to uniform even if the adversary gets to see another key for a ciphertext of its choice. We distinguish the following cases.

Case 1: The adversary corrupts P_η . For each session, the adversary can either reveal the session state or test this session. If the adversary reveals the state, we do not have to replace the session key. If the session is tested, the adversary does not know the state $E(s_\eta, \hat{sk})$ and thus we can replace the session key by ϵ -uniformity of Encap^* .

Case 2: The adversary does not corrupt P_η . In this case, we use that SE is IND-mRPA secure and replace \hat{sk} in the encrypted state with a random secret key for this party. Then we can use ϵ -uniformity to replace all tested keys for that party with random keys, as the state does not contain any information about \hat{sk} . After that, we have to switch back the state encryption to encrypt the real secret key \hat{sk} , getting ready for the next hybrid.

After these changes, the Test oracle will always output a random key, independent of the bit b .

Overall, the proof loses a factor of 2μ only in the IND-mRPA security of the symmetric encryption scheme. All other parts are tight.

Theorem 2 (Security of $\text{AKE}_{3\text{msg}}$ with Replay Attacks and without State Reveals). *For any adversary \mathcal{A} against $\text{AKE}_{3\text{msg}}$ with replay attacks and without state reveals, there exist an MU-EUF-CMA^{corr} adversary \mathcal{B}_{SIG} against SIG and an MUSC-otCCA adversary \mathcal{B}_{KEM} against KEM such that*

$$\begin{aligned} \text{Adv}_{\text{AKE}_{3\text{msg}}, \mu, \ell}^{\text{replay}}(\mathcal{A}) &\leq 2 \cdot \text{Adv}_{\text{KEM}, \mu \ell}^{\text{musc-otcca}}(\mathcal{B}_{\text{KEM}}) + 2 \cdot \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) \\ &\quad + 2(\mu \ell)^2 \cdot 2^{-\gamma} + \mu \ell^2 \cdot 2^{-\lambda} \end{aligned}$$

where γ is the diversity parameter of KEM and λ is the length of the nonce N in bits. Furthermore, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{KEM}})$ and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{SIG}})$.

Theorem 3 (Security of $\text{AKE}_{2\text{msg}}$ without State Reveals and Replay Attacks). *For any adversary \mathcal{A} against $\text{AKE}_{2\text{msg}}$ without state reveals and replay attacks, there exist an $\text{MU-EUF-CMA}^{\text{corr}}$ adversary \mathcal{B}_{SIG} against SIG and an MUC-otCCA adversary \mathcal{B}_{KEM} against KEM such that*

$$\text{Adv}_{\text{AKE}_{2\text{msg}}, \mu, \ell}(\mathcal{A}) \leq 2 \cdot \text{Adv}_{\text{KEM}, \mu, \ell}^{\text{muc-otcca}}(\mathcal{B}_{\text{KEM}}) + \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + (\mu\ell)^2 \cdot 2^{-\gamma},$$

where γ is the diversity parameter of KEM. Furthermore, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{KEM}})$ and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\text{SIG}})$.

The proofs of Theorem 2 and Theorem 3 are given in the full version [21], due to space limitations.

5 Signatures with Tight Adaptive Corruptions

5.1 Pairing Groups and MDDH Assumptions

Let GGen be a pairing group generation algorithm that returns a description $\mathcal{PG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, \mathcal{P}_1, \mathcal{P}_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order q for a λ -bit prime q , \mathcal{P}_1 and \mathcal{P}_2 are generators of \mathbb{G}_1 and \mathbb{G}_2 , respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficient computable (non-degenerated) bilinear map. $\mathcal{P}_T := e(\mathcal{P}_1, \mathcal{P}_2)$ is a generator in \mathbb{G}_T . In this paper, we only consider Type III pairings, where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficient homomorphism between them. All constructions in this paper can be easily instantiated with Type I pairings by setting $\mathbb{G}_1 = \mathbb{G}_2$ and defining the dimension k to be greater than 1.

We use the implicit representation of group elements as in [14]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_q$ define $[a]_s = a\mathcal{P}_s \in \mathbb{G}_s$ as the implicit representation of a in \mathbb{G}_s . Similarly, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n \times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of \mathbf{A} in \mathbb{G}_s . $\text{Span}(\mathbf{A}) := \{\mathbf{A}\mathbf{r} \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subset \mathbb{Z}_q^n$ denotes the linear span of \mathbf{A} , and similarly $\text{Span}([\mathbf{A}]_s) := \{[\mathbf{A}\mathbf{r}]_s \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subset \mathbb{G}_s^n$. Note that it is efficient to compute $[\mathbf{A}\mathbf{B}]_s$ given $([\mathbf{A}]_s, \mathbf{B})$ or $(\mathbf{A}, [\mathbf{B}]_s)$ with matching dimensions. We define $[\mathbf{A}]_1 \circ [\mathbf{B}]_2 := e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{A}\mathbf{B}]_T$, which can be efficiently computed given $[\mathbf{A}]_1$ and $[\mathbf{B}]_2$.

We recall the definition of the Matrix Decisional Diffie-Hellman (MDDH) and related assumptions from [14].

Definition 12 (Matrix distribution). *Let $k, \ell \in \mathbb{N}$ with $\ell > k$. We call $\mathcal{D}_{\ell, k}$ a matrix distribution if it outputs matrices in $\mathbb{Z}_q^{\ell \times k}$ of full rank k in polynomial time. Let $\mathcal{D}_k := \mathcal{D}_{k+1, k}$.*

For positive integers $k, \eta, n \in \mathbb{N}^+$ and a matrix $\mathbf{A} \in \mathbb{Z}_q^{(k+\eta) \times n}$, we denote the k rows of \mathbf{A} by $\overline{\mathbf{A}} \in \mathbb{Z}_q^{k \times n}$ and the lower η rows of \mathbf{A} by $\underline{\mathbf{A}} \in \mathbb{Z}_q^{\eta \times n}$. Without loss of generality, we assume $\overline{\mathbf{A}}$ for $\mathbf{A} \leftarrow_s \mathcal{D}_{\ell, k}$ form an invertible square matrix in $\mathbb{Z}_q^{k \times k}$. The $\mathcal{D}_{\ell, k}$ -MDDH problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{A}\mathbf{w}])$ and $([\mathbf{A}], [\mathbf{u}])$ where $\mathbf{A} \leftarrow_s \mathcal{D}_{\ell, k}$, $\mathbf{w} \leftarrow_s \mathbb{Z}_q^k$ and $\mathbf{u} \leftarrow_s \mathbb{Z}_q^\ell$.

Definition 13 ($\mathcal{D}_{\ell,k}$ -MDDH assumption). Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_{\ell,k}$ -MDDH assumption holds relative to GGen in group \mathbb{G}_s if for all adversaries \mathcal{A} , it holds that

$$\text{Adv}_{\text{GGen}, \mathcal{D}_{\ell,k}, \mathbb{G}_s}^{\text{MDDH}}(\mathcal{A}) := |\Pr[\mathcal{A}(\mathcal{P}\mathcal{G}, [\mathbf{A}]_s, [\mathbf{A}\mathbf{w}]_s) \Rightarrow 1] - \Pr[\mathcal{A}(\mathcal{P}\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s) \Rightarrow 1]|$$

is negligible where the probability is taken over $\mathcal{P}\mathcal{G} \leftarrow_s \text{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow_s \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow_s \mathbb{Z}_q^k$ and $\mathbf{u} \leftarrow_s \mathbb{Z}_q^\ell$.

Definition 14 (Uniform distribution). Let $k, \ell \in \mathbb{N}^+$ with $\ell > k$. We call $\mathcal{U}_{\ell,k}$ a uniform distribution if it outputs uniformly random matrices in $\mathbb{Z}_q^{\ell \times k}$ of rank k in polynomial time. Let $\mathcal{U}_k := \mathcal{U}_{k+1,k}$.

Lemma 1 ($\mathcal{D}_{\ell,k}$ -MDDH $\Rightarrow \mathcal{U}_k$ -MDDH [14]). Let $\ell, k \in \mathbb{N}_+$ with $\ell > k$ and let $\mathcal{D}_{\ell,k}$ be a matrix distribution. A \mathcal{U}_k -MDDH instance is at least as hard as an $\mathcal{D}_{\ell,k}$ instance. More precisely, for each adversary \mathcal{A} there exists an adversary \mathcal{B} with

$$\text{Adv}_{\text{GGen}, \mathcal{U}_k, \mathbb{G}_s}^{\text{MDDH}}(\mathcal{A}) \leq \text{Adv}_{\text{GGen}, \mathcal{D}_{\ell,k}, \mathbb{G}_s}^{\text{MDDH}}(\mathcal{B})$$

and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$.

The Kernel-Diffie-Hellman assumption (\mathcal{D}_k -KMDH) [31] is a (weaker) computational analogue of the \mathcal{D}_k -MDDH Assumption.

Definition 15 (\mathcal{D}_k -KMDH). Let \mathcal{D}_k be a matrix distribution. We say that the \mathcal{D}_k -Kernel Diffie-Hellman (\mathcal{D}_k -KMDH) assumption holds relative to a prime order group \mathbb{G}_s for $s \in \{1, 2\}$ if for all PPT adversaries \mathcal{A} ,

$$\text{Adv}_{\text{GGen}, \mathcal{D}_k, \mathbb{G}_s}^{\text{KMDH}}(\mathcal{A}) := \Pr[\mathbf{c}^\top \mathbf{A} = \mathbf{0} \wedge \mathbf{c} \neq \mathbf{0} \mid [\mathbf{c}]_{3-s} \leftarrow_s \mathcal{A}(\mathcal{P}\mathcal{G}, [\mathbf{A}]_s)],$$

where the probabilities are taken over $\mathcal{P}\mathcal{G} \leftarrow_s \text{GGen}(1^\lambda)$ and $\mathbf{A} \leftarrow_s \mathcal{D}_k$.

5.2 Previous Schemes with Tight Adaptive Corruptions

We will construct a signature scheme with tight MU-EUF-CMA^{corr} security and only small constant number of elements in signatures. Such a scheme has been proposed in [2, Section 2.3] (called SIG_C), but we identify a gap in their proof. We now present the gap in their security proof and why we think it will be hard to close it.

The construction of SIG_C follows the BKP IBE schemes [6], namely, it tightly transforms an affine MAC [6] into a signature in the multi-user setting. In order to have a tightly MU-EUF-CMA^{corr} secure signature scheme, the underlying MAC needs to be tightly secure against adaptive corruption of its secret keys in the multi-user setting. We will now point to potential problems in formally proving it.

We abstract the underlying MAC of SIG_C as MAC_{BHJKL}: For message space $\{0, 1\}^\ell$, it chooses $\mathbf{A}' \leftarrow_s \mathcal{D}_k$ and random vectors $\mathbf{x}_{i,j} \leftarrow_s \mathbb{Z}_q^k$ (for $1 \leq i \leq \ell$ and $j = 0, 1$). Then it defines $\mathbf{B} := \overline{\mathbf{A}'} \in \mathbb{Z}_q^{k \times k}$ and publishes system parameters

$\text{pp} := ([\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \leq i \leq \ell, j=0,1})$. For each user n , it chooses its MAC secret key as $[x'_n]_1 \leftarrow_{\$} \mathbb{G}_1$, and its MAC tag consist of $([\mathbf{t}]_1, [u]_1)$, where

$$\begin{aligned} \mathbf{t} &= \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^k \quad \text{for } \mathbf{s} \leftarrow_{\$} \mathbb{Z}_q^k \\ u &= x'_n + \mathbf{t}^\top \underbrace{\sum_i \mathbf{x}_{i,m_i}}_{=: \mathbf{x}(m)} \in \mathbb{Z}_q. \end{aligned} \tag{3}$$

In their security proof, they argue that $[u]_1$ in the MAC tagging queries is pseudo-random, given pp and some of the secret keys $[x'_n]_1$ (via the adaptive corruption queries) to an adversary.⁷ In achieving this, they define a sequence of hybrids H_j for $1 \leq j \leq \ell$. In each H_j , they replace x'_n for each user n with $\text{RF}_{n,j}(m_{|j})$, where $\text{RF}_{n,j} : \{0, 1\}^j \rightarrow \mathbb{Z}_q$ is a random function and m is the first tagging query to user n , and generate the MAC tag of m' as

$$u = \text{RF}_{n,j}(m'_{|j}) + \mathbf{t}^\top \mathbf{x}(m') \tag{4}$$

with \mathbf{t} as in Eq. (3).

In their final step (between H_ℓ and GAME 4), they argue that the distribution of $u = \text{RF}_{n,\ell}(m') + \mathbf{t}^\top \mathbf{x}(m')$ is uniformly random (as in GAME 4) even for an unbounded adversary, given pp and adaptive corruptions. Then they conclude that H_ℓ (where $u = \text{RF}_{n,\ell}(m') + \mathbf{t}^\top \mathbf{x}(m')$) and GAME 4 (where u is chosen uniformly at random) are identical and $\Pr[\chi_4] = \Pr[H_\ell = 1]$ (according to their notation). However, this is not the case: $\mathbf{B} \in \mathbb{Z}_q^{k \times k}$ is full-rank and thus, given $[\mathbf{B}^\top \mathbf{x}_{i,j}]_1$ in pp , $\mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ is uniquely defined. (For concreteness, imagine a simple example where an (unbounded) adversary \mathcal{A} only queries one MAG tag for message m for user n and then asks for the secret key $[x'_n]_1 := \text{RF}_{n,\ell}(m)$ of user n . Then, \mathcal{A} sees that $u = \text{RF}_{n,\ell}(m) + \mathbf{t}^\top \mathbf{x}(m)$ is uniquely defined by $[x'_n]_1, [\mathbf{t}]_1$ and pp in H_ℓ , while u is uniformly at random in GAME 4.) We suppose this gap is inherent, since the terms $\mathbf{B}^\top \mathbf{x}_{i,j}$ completely leak the information about $\mathbf{x}_{i,j}$. This is also the same reason why the BKP MAC cannot be used to construct a tightly secure hierarchical IBE (HIBE) (cf. [26] for more discussion).

To resolve this, we follow the tightly secure HIBE approach in [26] and choose $\mathbf{B} \leftarrow_{\$} \mathbb{Z}_q^{3k \times k}$. Now, there is a non-zero kernel matrix $\mathbf{B}^\perp \in \mathbb{Z}_q^{3k \times 2k}$ for \mathbf{B} (with overwhelming probability), and the mapping $\mathbf{x}_{i,j} \in \mathbb{Z}_q^{3k} \mapsto \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ is lossy. In particular, the information about $\mathbf{x}_{i,j}$ in the orthogonal space of \mathbf{B} is perfectly hidden from (unbounded) adversaries, given $\mathbf{B}^\top \mathbf{x}_{i,j}$.

5.3 Our Construction

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ be a function chosen from a collision-resistant hash function family \mathcal{H} . Our signature scheme $\text{SIG}_{\text{MDDH}} := (\text{SIG.Setup}, \text{SIG.Gen}, \text{Sign}, \text{Ver})$ is defined in Fig. 6. Correctness can be verified as

$$[\mathbf{v}, u]_1 \circ [\mathbf{A}]_2 = [(y', x') \cdot \mathbf{A} + \mathbf{t}^\top \cdot (\mathbf{Y}(\text{hm}) \mid \mathbf{x}(\text{hm})) \cdot \mathbf{A}]_T$$

for $([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1) \leftarrow_{\$} \text{Sign}(ssk, m)$.

⁷ This is different to the BKP IBE where $[\mathbf{B}^\top \mathbf{x}_{i,j}]_1$ and $[x'_n]_1$ are not available to an adversary.

<p>SIG.Setup: $\mathcal{PG} \leftarrow \text{GGen}$ $\mathbf{A} \leftarrow \mathcal{D}_k; \mathbf{B} \leftarrow \mathcal{U}_{3k,k}$ For $1 \leq i \leq \lambda$ and $j = 0, 1$: $\mathbf{x}_{i,j} \leftarrow \mathbb{Z}_q^{3k}; \mathbf{Y}_{i,j} \leftarrow \mathbb{Z}_q^{3k \times k}$ $\mathbf{Z}_{i,j} := (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \cdot \mathbf{A} \in \mathbb{Z}_q^{3k \times (k+1)}$ $\mathbf{P}_{i,j} := \mathbf{B}^\top \cdot (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \in \mathbb{Z}_q^{k \times (k+1)}$ $\text{pp} := (\mathcal{PG}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ Return pp</p> <p>SIG.Gen(pp) $x' \leftarrow \mathbb{Z}_q; \mathbf{y}' \leftarrow \mathbb{Z}_q^{1 \times k}$ $ssk := ([x']_1, [\mathbf{y}']_1)$ $vk := [z']_2 := [(y' \parallel x')\mathbf{A}]_2 \in \mathbb{G}_2^{1 \times k}$ Return (vk, ssk)</p>	<p>Sign(ssk, m): $\mathbf{s} \leftarrow \mathbb{Z}_q^k; \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}$ $\text{hm} := H(vk, m)$ $u := x' + \mathbf{s}^\top \mathbf{B}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q$ $\mathbf{v} := \mathbf{y}' + \mathbf{s}^\top \mathbf{B}^\top \mathbf{Y}(\text{hm}) \in \mathbb{Z}_q^{1 \times k}$ Return $\sigma := ([t]_1, [u]_1, [\mathbf{v}]_1)$</p> <p>Ver(vk, m, σ := $([t]_1, [u]_1, [\mathbf{v}]_1)$): $\text{hm} := H(vk, m)$ If $[\mathbf{v}, u]_1 \circ [\mathbf{A}]_2 = [1]_1 \circ [z']_2 + [t]_1^\top \circ [\mathbf{Z}(\text{hm})]_2$: Return 1 Else: Return 0</p>
---	--

Fig. 6. Our signature scheme with tight adaptive corruptions, where for $\text{hm} \in \{0, 1\}^\lambda$ we define the functions $\mathbf{x}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{x}_{i,\text{hm}_i}$, $\mathbf{Y}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{Y}_{i,\text{hm}_i}$, $\mathbf{Z}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{Z}_{i,\text{hm}_i}$, and $\mathbf{P}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{P}_{i,\text{hm}_i}$.

Theorem 4 (Security of SIG_{MDDH}). *For any adversary \mathcal{A} against the MU-EUF-CMA^{corr} security of SIG_{MDDH}, there are adversaries \mathcal{B} against the collision resistance of \mathcal{H} , \mathcal{B}_1 against the $\mathcal{U}_{3k,k}$ -MDDH assumption over \mathbb{G}_1 and \mathcal{B}_2 against the \mathcal{D}_k -KMDH assumption over \mathbb{G}_2 with*

$$\Pr[\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}} \Rightarrow 1] \leq \text{Adv}_{\mathcal{H}}^{\text{cr}}(\mathcal{B}) + (8k\lambda + 2k) \text{Adv}_{\text{GGen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\text{MDDH}}(\mathcal{B}_1) + \text{Adv}_{\text{GGen}, \mathcal{D}_k, \mathbb{G}_2}^{\text{KMDH}}(\mathcal{B}_2) + \frac{\lambda + 2k + 2}{q - 1},$$

where $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_1) \approx \mathbf{T}(\mathcal{B}_2)$.

Proof. We prove the tight MU-EUF-CMA^{corr} security of SIG_{MDDH} with a sequence of games given in Fig. 7. Let \mathcal{A} be an adversary against the MU-EUF-CMA^{corr} security of SIG_{MDDH}, and let Win_i denote the probability that \mathcal{G}_i returns 1.

Game \mathcal{G}_0 : \mathcal{G}_0 is the original MU-EUF-CMA^{corr} security experiment $\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}}$ (see the full version [21] for the formal definition). In addition to the original game, we add a rejection rule if there is a collision between the forgery and a signing query, namely, $H(vk_{i^*}, m^*) = H(vk_i, m)$ where (i, m) is one of the signing queries. By the collision resistance of H , we have

$$|\Pr[\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}} \Rightarrow 1] - \Pr[\text{Win}_0]| \leq \text{Adv}_{\mathcal{H}}^{\text{cr}}(\mathcal{B}).$$

For better readability, we assume all the signing queries are distinct for the following games. If the same (i, m) is asked multiple times, we can take the first response $([t]_1, [u]_1, [\mathbf{v}]_1)$ and answer the repeated queries with the re-randomization $([t']_1, [u']_1, [\mathbf{v}']_1)$ as $\mathbf{t}' := \mathbf{t} + \mathbf{B}\mathbf{s}'$ (for $\mathbf{s}' \leftarrow \mathbb{Z}_q^k$), $u' := u + \mathbf{s}'^\top (\mathbf{B}^\top \mathbf{x}(\text{hm}))$ and $\mathbf{v}' := \mathbf{v} + \mathbf{s}'^\top (\mathbf{B}^\top \mathbf{Y}(\text{hm}))$ and $\text{hm} := H(vk_i, m)$. Note that this will not change the view of \mathcal{A} .

$\mathbb{G}_0, \boxed{\mathbb{G}_1, \mathbb{G}_2};$ $\mathcal{PG} \leftarrow \mathcal{GGen}; \mathbf{A} \leftarrow \mathcal{D}_k; \mathbf{B} \leftarrow \mathcal{U}_{3k,k}$ <p>For $1 \leq i \leq \lambda$ and $j = 0, 1$:</p> $\mathbf{x}_{i,j} \leftarrow \mathbb{Z}_q^{3k}; \mathbf{Y}_{i,j} \leftarrow \mathbb{Z}_q^{3k \times k}$ $\mathbf{Z}_{i,j} := (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \cdot \mathbf{A} \in \mathbb{Z}_q^{3k \times k}$ $\mathbf{P}_{i,j} := \mathbf{B}^\top \cdot (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \in \mathbb{Z}_q^{k \times (k+1)}$ <div style="border: 1px solid black; padding: 2px; margin: 5px 0;"> $\mathbf{Z}_{i,j} \leftarrow \mathbb{Z}_q^{3k \times k}$ $\mathbf{d}_{i,j} := \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ $\mathbf{E}_{i,j} := (\mathbf{B}^\top \mathbf{Z}_{i,j} - \mathbf{d}_{i,j} \cdot \underline{\mathbf{A}}) \overline{\mathbf{A}}^{-1} \in \mathbb{Z}_q^{k \times k}$ $\mathbf{P}_{i,j} := (\mathbf{E}_{i,j} \parallel \mathbf{d}_{i,j})$ </div> $\text{pp} := (\mathcal{PG}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ <p>For $1 \leq i \leq \mu$:</p> $x'_i \leftarrow \mathbb{Z}_q; \mathbf{y}'_i \leftarrow \mathbb{Z}_q^{1 \times k}$ $\mathbf{z}'_i := (\mathbf{y}'_i \parallel x'_i) \mathbf{A} \in \mathbb{Z}_q^{1 \times k}$ <div style="border: 1px solid black; padding: 2px; margin: 5px 0;"> $\mathbf{z}'_i \leftarrow \mathbb{Z}_q^{1 \times k}; \mathbf{y}'_i := (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}}) \overline{\mathbf{A}}^{-1}$ </div> $\text{ssk}_i := ([x'_i]_1, [\mathbf{y}'_i]_1)$ $vk_i := [\mathbf{z}'_i]_2$ $(i^*, \mathbf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SIGN}}(\cdot, \cdot), \mathcal{O}_{\text{CORR}}(\cdot)}(\text{pp}, \{vk_i\}_{1 \leq i \leq \mu})$ <p>If $(i^* \in \mathcal{S}^{\text{corr}}) \vee (m^* \in \mathcal{M}_{i^*}) \vee (\text{Ver}(vk_{i^*}, m^*, \sigma^*) = 0)$:</p> <p style="padding-left: 20px;">Return 0</p> $\text{hm}^* := H(vk_{i^*}, \mathbf{m}^*)$ <p>If $\exists 1 \leq i \leq \mu \wedge m \in \mathcal{M}_i : H(vk_i, m) = \text{hm}^*$</p> <p style="padding-left: 20px;">Return 0</p> <div style="border: 1px dashed black; padding: 2px; margin: 5px 0;"> $\text{Parse } \sigma^* := ([\mathbf{t}^*]_1, [u^*]_1, [\mathbf{v}^*]_1)$ $\text{If } [u^*]_1 \neq [x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\text{hm}^*)$ <p style="padding-left: 20px;">Return 0</p> </div> <p>Return 1</p>	$\mathcal{O}_{\text{SIGN}}(i, \mathbf{m}):$ $\mathbf{s} \leftarrow \mathbb{Z}_q^k; \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}$ $\text{hm} := H(vk_i, \mathbf{m})$ $u := x'_i + \mathbf{s}^\top \mathbf{B}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q$ $\mathbf{v} := \mathbf{y}'_i + \mathbf{s}^\top \mathbf{B}^\top \mathbf{Y}(\text{hm}) \in \mathbb{Z}_q^{1 \times k}$ <div style="border: 1px solid black; padding: 2px; margin: 5px 0;"> $\mathbf{v} := (\mathbf{z}'_i + \mathbf{t}^\top \mathbf{Z}(\text{hm}) - u \cdot \underline{\mathbf{A}}) \cdot \overline{\mathbf{A}}^{-1}$ </div> $\mathcal{M}_i := \mathcal{M}_i \cup \{m\}$ <p>Return $\sigma := ([t]_1, [u]_1, [\mathbf{v}]_1)$</p> $\mathcal{O}_{\text{CORR}}(i):$ $\mathcal{S}^{\text{corr}} := \mathcal{S}^{\text{corr}} \cup \{i\}$ <p>Return ssk_i</p>
---	---

Fig. 7. Games used to prove Theorem 4.

Game \mathbb{G}_1 : For verifying the forgery, in addition to using Ver , we use the secret $[x'_{i^*}]_1$ and $([x_{j,b}]_1)_{1 \leq j \leq \lambda}$ to check if $([\mathbf{t}^*]_1, [u^*]_1)$ in the forgery satisfies the following equation:

$$[u^*]_1 = [x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\text{hm}^*). \quad (5)$$

We note that

$$\begin{aligned} \text{Ver}(vk_{i^*}, m^*, \sigma^*) &= 1 \\ \Leftrightarrow (\mathbf{v} \parallel u) \cdot \mathbf{A} &= (\mathbf{y}'_{i^*} \parallel x'_{i^*}) \mathbf{A} + \mathbf{t}^{*\top} \cdot (\mathbf{Y}(\text{hm}) \parallel \mathbf{x}(\text{hm}^*)) \cdot \mathbf{A}. \end{aligned}$$

Thus, if Eq. (5) does not hold, then the vector $[(\mathbf{v} \parallel u)]_1 - ([\mathbf{y}'_{i^*} \parallel x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\text{hm}^*)) \in \mathbb{G}_1^{1 \times (k+1)}$ is non-zero and orthogonal to $[\mathbf{A}]_2$. Therefore, we bound the difference between \mathbb{G}_0 and \mathbb{G}_1 with the \mathcal{D}_k -KMDH assumption as

$$|\Pr[\text{Win}_0] - \Pr[\text{Win}_1]| \leq \text{Adv}_{\mathbb{GGen}, \mathcal{D}_k, \mathbb{G}_2}^{\text{KMDH}}(\mathcal{B}).$$

Game G_2 : We do not use the values $\mathbf{Y}_{j,b}$ (for $1 \leq j \leq \lambda$ and $b = 0, 1$) and \mathbf{y}'_i (for $1 \leq i \leq \mu$) to simulate G_2 . We make this change by substituting all $\mathbf{Y}_{j,b}$ and \mathbf{y}'_i using the formulas

$$\mathbf{Y}_{j,b}^\top = (\mathbf{Z}_{j,b} - \mathbf{x}_{j,b} \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1} \text{ and } \mathbf{y}'_i = (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1}, \quad (6)$$

respectively. More precisely, the public parameters \mathbf{pp} are computed by picking $\mathbf{Z}_{j,b}$ and $\mathbf{x}_{j,b}$ at random and then defining $\mathbf{Y}_{j,b}$ using Eq. (6). The verification keys vk_i for user i ($1 \leq i \leq \mu$) are computed by picking \mathbf{z}'_i and x'_i at random. For $\mathcal{O}_{\text{SIGN}}(i, \mathbf{m})$, we now compute

$$\begin{aligned} \mathbf{v} &:= \mathbf{y}'_i + \mathbf{t}^\top \mathbf{Y}(\text{hm}) \in \mathbb{Z}_q^{1 \times k} \\ &= (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1} + \mathbf{t}^\top (\mathbf{Z}(\text{hm}) - \mathbf{x}(\text{hm}) \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1} \\ &= (\mathbf{z}'_i + \underbrace{\mathbf{t}^\top \mathbf{Z}(\text{hm}) - (x'_i + \mathbf{t}^\top \mathbf{x}(\text{hm})) \cdot \underline{\mathbf{A}}}_{=u})(\overline{\mathbf{A}})^{-1}. \end{aligned}$$

The secret verification of the forgery can be done by knowing x'_{i^*} and $\mathbf{x}_{j,b}$.

The changes in G_2 are only conceptual, since Eq. (6) are equivalent to $\mathbf{Z}_{j,b} = (\mathbf{Y}_{j,b} \parallel \mathbf{x}_{j,b})\mathbf{A}$ and $\mathbf{z}'_i = (\mathbf{y}'_i \parallel x'_i)\mathbf{A}$. Thus, we have

$$\Pr[\text{Win}_1] = \Pr[\text{Win}_2].$$

In order to bound $\Pr[\text{Win}_2]$, consider a “message authentication code” MAC which is defined as follows.

- The public parameters consist of $\mathbf{pp}_{\text{MAC}} := (\mathcal{P}\mathcal{G}, [\mathbf{B}]_1, ([\mathbf{d}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$, where $\mathbf{d}_{i,j} := \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ for $\mathbf{x}_{i,j} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{3k}$ and $\mathbf{B} \leftarrow_{\mathcal{S}} \mathcal{U}_{3k,k}$.
- The secret key is $[x']_1$.
- The MAC tag on hm is $([\mathbf{t}]_1, [u]_1)$, where $\mathbf{t} := \mathbf{B}\mathbf{s}$ and $u := x' + \mathbf{t}^\top \mathbf{x}(\text{hm})$, for $\mathbf{s} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$.

Note that strictly speaking MAC is not a MAC since verification cannot only be done efficiently by knowing the values $\mathbf{x}_{i,j}$.

The following lemma states MU-EUF-CMA^{corr} security of MAC, with proof in the full version [21].

Lemma 2 (Core Lemma). *For every adversaries \mathcal{A} interacting with UF-CMA^{corr}, there exists an adversary \mathcal{B} against the $\mathcal{U}_{3k,k}$ -MDDH assumption in \mathbb{G}_1 with*

$$\Pr[\text{UF-CMA}_{\mathcal{A}}^{\text{corr}} \Rightarrow 1] \leq (8k\lambda + 2k) \cdot \text{Adv}_{\text{GGen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\text{MDDH}}(\mathcal{B}_1) + \frac{4\lambda + 2k + 2}{q - 1},$$

and $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$, where Q_e is the number of \mathcal{A} 's queries to \mathcal{O}_{MAC} .

Finally, we bound the probability that the adversary wins in G_2 using our Core Lemma (Lemma 2) by constructing an adversary \mathcal{B}_{MAC} as in Fig. 9.

$$\Pr[\text{Win}_2] = \Pr[\text{UF-CMA}_{\mathcal{B}_{\text{MAC}}}^{\text{corr}} \Rightarrow 1].$$

$\overline{\mathcal{A}}^{\text{UF-CMA}^{\text{corr}}}$ $\beta = 0$ $\mathcal{P}\mathcal{G} \leftarrow_s \text{GGen}$ $\mathbf{B} \leftarrow_s \mathcal{U}_{3k,k}$ <p>For $1 \leq i \leq \lambda$ and $j = 0, 1$:</p> $\mathbf{x}_{i,j} \leftarrow_s \mathbb{Z}_q^{3k}$ $\text{pp}_{\text{MAC}} := (\mathcal{P}\mathcal{G}, [\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ <p>For $1 \leq i \leq \mu$:</p> $x_i \leftarrow_s \mathbb{Z}_q$ $\mathcal{A}^{\mathcal{O}_{\text{MAC}}(\cdot), \mathcal{O}_{\text{Ver}}(\cdot, \cdot), \mathcal{O}'_{\text{CORR}}(\cdot)}(\text{pp}_{\text{MAC}})$ <p>Return β</p>	$\mathcal{O}_{\text{MAC}}(i, \text{hm}):$ $\mathcal{Q} := \mathcal{Q} \cup \{(i, \text{hm})\}$ $\mathbf{s} \leftarrow_s \mathbb{Z}_q^k; \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}$ $u := x'_i + \mathbf{t}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q$ <p>Return $\sigma := ([\mathbf{t}]_1, [u]_1)$</p> $\mathcal{O}_{\text{Ver}}(i^*, \text{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1)): \text{// at most once}$ <p>If $(i^*, \text{hm}^*) \in \mathcal{Q} \vee (i^* \in \mathcal{L})$:</p> <p>Return 0</p> <p>If $[u^*]_1 := [x'_{i^*}]_1 + [\mathbf{t}^{*\top}]_1 \cdot \mathbf{x}(\text{hm}^*)$:</p> <p>$\beta := 1$</p> <p>Return 1</p> <p>Else: Return 0</p> $\mathcal{O}'_{\text{CORR}}(i)$ $\mathcal{L} := \mathcal{L} \cup \{i\}$ <p>Return $[x'_i]_1$</p>
---	---

Fig. 8. Game $\text{UF-CMA}^{\text{corr}}$ for Lemma 2.

In order to analyze $\text{Pr}[\text{Win}_2]$ we argue as follows. The simulated pp and $(vk_i)_{1 \leq i \leq \mu}$ are distributed as in G_2 . Further, queries to $\mathcal{O}_{\text{SIGN}}$ and $\mathcal{O}_{\text{CORR}}$ from ssk_i can be perfectly simulated using \mathcal{O}_{MAC} and $\mathcal{O}'_{\text{CORR}}$, respectively. The additional group elements $[v]_1$ from σ and $[y'_i]_1$ can be simulated as in G_2 . Finally, using a valid forgery (i^*, m^*, σ^*) output by \mathcal{A} , \mathcal{B}_{MAC} wins its own game by calling $\mathcal{O}_{\text{Ver}}(i^*, \text{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1))$, where $([\mathbf{t}^*]_1, [u^*]_1)$ is a valid MAC tag on hm^* for user i^* . □

$\overline{\mathcal{B}}^{\mathcal{O}_{\text{MAC}}(\cdot), \mathcal{O}_{\text{Ver}}(\cdot, \cdot), \mathcal{O}'_{\text{CORR}}(\cdot)}(\text{pp}_{\text{MAC}}):$ <p>Parse $\text{pp}_{\text{MAC}} =: (\mathcal{P}\mathcal{G}, [\mathbf{B}]_1, ([\mathbf{d}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$</p> $\mathbf{A} \leftarrow_s \mathcal{D}_k$ <p>For $1 \leq i \leq \lambda$ and $j = 0, 1$:</p> $\mathbf{Z}_{i,j} \leftarrow_s \mathbb{Z}_q^{3k \times k}$ $\mathbf{E}_{i,j} := (\mathbf{B}^\top \mathbf{Z}_{i,j} - \mathbf{d}_{i,j} \cdot \mathbf{A}) \overline{\mathbf{A}}^{-1} \in \mathbb{Z}_q^{k \times k}$ $\mathbf{P}_{i,j} := (\mathbf{E}_{i,j} \parallel \mathbf{d}_{i,j})$ $\text{pp} := (\mathcal{P}\mathcal{G}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ <p>For $1 \leq i \leq \mu$:</p> $\mathbf{z}'_i \leftarrow_s \mathbb{Z}_q^{1 \times k}$ $vk_i := [\mathbf{z}'_i]_2 \quad \text{// ssk}_i \text{ is undefined}$ $(i^*, m^*, \sigma^*) \leftarrow_s \mathcal{A}^{\mathcal{O}_{\text{SIGN}}(\cdot, \cdot), \mathcal{O}_{\text{CORR}}(\cdot)}(\text{pp}, \{vk_i\}_{1 \leq i \leq \mu})$ <p>If $(i^* \in \mathcal{S}^{\text{corr}}) \vee (m^* \in \mathcal{M}_{i^*}) \vee (\text{Ver}(vk_{i^*}, m^*, \sigma^*) = 0)$:</p> <p>Return 0</p> $\text{hm}^* := H(vk_{i^*}, m^*)$ <p>If $\exists 1 \leq i \leq \mu \wedge m \in \mathcal{M}_i : H(vk_i, m) = \text{hm}^*$</p> <p>Return 0</p> <p>Parse $\sigma^* := ([\mathbf{t}^*]_1, [u^*]_1, [v^*]_1)$</p> $\mathcal{O}_{\text{Ver}}(i^*, \text{hm}^*, [\mathbf{t}^*]_1, [u^*]_1)$ <p>Return 1</p>	$\mathcal{O}_{\text{SIGN}}(i, m):$ $\text{hm} := H(vk_i, m)$ $([\mathbf{t}]_1, [u]_1) \leftarrow_s \mathcal{O}_{\text{MAC}}(\text{hm})$ $\mathbf{v} := (\mathbf{z}'_i + \mathbf{t}^\top \mathbf{Z}(\text{hm}) - u \cdot \mathbf{A}) \cdot (\overline{\mathbf{A}})^{-1}$ $\mathcal{M}_i := \mathcal{M}_i \cup \{m\}$ <p>Return $\sigma := ([\mathbf{t}]_1, [u]_1, [v]_1)$</p> $\mathcal{O}'_{\text{CORR}}(i):$ $\mathcal{S}^{\text{corr}} := \mathcal{S}^{\text{corr}} \cup \{i\}$ $[x'_i]_1 \leftarrow \mathcal{O}'_{\text{CORR}}(i)$ $y'_i := (\mathbf{z}'_i - x'_i \cdot \mathbf{A}) (\overline{\mathbf{A}})^{-1}$ <p>Return $\text{ssk}_i := ([x'_i]_1, [y'_i]_1)$</p>
---	--

Fig. 9. Reduction \mathcal{B}_{MAC} to bound the winning probability in G_2 . \mathcal{B}_{MAC} receives pp_{MAC} and gets oracle access to \mathcal{O}_{MAC} and \mathcal{O}_{Ver} , and $\mathcal{O}'_{\text{CORR}}$ as in Fig. 8.

6 Concrete Instantiation of Our AKE Protocols

For $\text{AKE}_{3\text{msg}}$, we use our new signature scheme SIG_{MDDH} (Fig. 6) and the ϵ -MU-SIM KEM constructed from the MDDH-based hash proof system HPS_{MDDH} (cf. the full version [21]). For $\text{AKE}_{3\text{msg}}^{\text{state}}$, the symmetric encryption scheme to protect against state reveals can be instantiated using any weakly secure (deterministic) encryption scheme such as AES or even a weak PRF.

For the KEM constructed in the full version [21], the KEM public key consists of $2k$ group elements and the ciphertext of $k + 1$ group elements. A signature consists of $4k + 1$ group elements, cf. Fig. 6. Therefore, the first message is a bitstring of length λ , the second message consists of $6k + 1$ group elements and the third message consists of $5k + 2$ group elements. For $k = 1$, we get an efficient SXDH-based scheme with 15 elements in total.

We instantiate protocol $\text{AKE}_{2\text{msg}}$ using our signature scheme from Fig. 6 and the MUC-otCCA secure KEM from Han *et al.* [22]. γ -diversity of the KEM is proven in [29, Appendix D.2]. We analyze the communication complexity of $\text{AKE}_{2\text{msg}}$ as follows. The KEM public key consists of $k^2 + 3k$ group elements and the ciphertext of $2k + 3$ group elements. A signature consists of $4k + 1$ group elements. Therefore, the first message consists of $k^2 + 7k + 1$ group elements and the second message consists of $6k + 4$ group elements. For $k = 1$, we get an efficient SXDH-based scheme with $9 + 10 = 19$ group elements in total.

For an overview we refer to Table 1 of the introduction.

Acknowledgments. We would like to thank the reviewers for their helpful comments. Shuai Han and Shengli Liu were partially supported by National Natural Science Foundation of China (Grant Nos. 61925207, 62002223), Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), Shanghai Sailing Program (20YF1421100), Young Elite Scientists Sponsorship Program by China Association for Science and Technology, and the National Key Research and Development Project 2020YFA0712300. Tibor Jager was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823. Eike Kiltz was supported by the BMBF iBlockchain project, the EU H2020 PROMETHEUS project 780701, DFG SPP 1736 Big Data, and the DFG Cluster of Excellence 2092 CASA. Doreen Riepel was supported by the Deutsche Forschungsgemeinschaft (DFG) Cluster of Excellence 2092 CASA. Sven Schäge was supported by the German Federal Ministry of Education and Research (BMBF), Project DigiSeal (16KIS0695) and Huawei Technologies Düsseldorf, Project vHSM.

References

1. Bader, C.: Efficient signatures with tight real world security in the random-oracle model. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 370–383. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12280-9_24

2. Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_26
3. Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_10
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 1993, pp. 62–73. ACM Press, November 1993
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_21
6. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_23
7. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_28
8. Cramer, R., et al.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_31
9. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
10. Cremers, C.J.F., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 734–751. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_42
11. Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. ACNS 2021 (2021). <https://eprint.iacr.org/2020/1029>
12. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: 24th International Conference on Practice and Theory of Public-Key Cryptography, PKC 2021 (2021)
13. Diemert, D., Jager, T.: On the tight security of TLS 1.3: theoretically-sound cryptographic parameters for real-world deployments. Cryptology ePrint Archive, Report 2020/726 (2020). <https://eprint.iacr.org/2020/726>
14. 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 (2013). https://doi.org/10.1007/978-3-642-40084-1_8
15. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for Diffie-Hellman assumptions. J. Cryptol. **30**(1), 242–288 (2017)
16. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_28

17. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_1
18. Gay, R., Hofheinz, D., Kohl, L.: Kurosawa-Desmedt meets tight security. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 133–160. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_5
19. 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, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_4
20. Günther, C.G.: An identity-based key-exchange protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_5
21. Han, S., et al.: Authenticated key exchange and signatures with tight security in the standard model. Cryptology ePrint Archive, Report 2021/863 (2021). <https://eprint.iacr.org/2021/863>
22. Han, S., Liu, S., Lyu, L., Gu, D.: Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 417–447. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_15
23. 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 (2021). https://doi.org/10.1007/978-3-030-77870-5_5
24. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_17
25. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
26. Langrehr, R., Pan, J.: Tightly secure hierarchical identity-based encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part I. LNCS, vol. 11442, pp. 436–465. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_15
27. Langrehr, R., Pan, J.: Unbounded HIBE with tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 129–159. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_5
28. Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: defining trivial attacks for security protocols is not trivial. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 1343–1360. ACM Press, October/November 2017
29. 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, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_27
30. Morgan, A., Pass, R., Shi, E.: On the adaptive security of MACs and PRFs. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 724–753. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_24

31. Morillo, P., Ràfols, C., Villar, J.L.: The kernel matrix Diffie-Hellman assumption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 729–758. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_27
32. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_8



KHAPE: Asymmetric PAKE from Key-Hiding Key Exchange

Yanqi Gu¹, Stanislaw Jarecki^{1(✉)}, and Hugo Krawczyk²

¹ University of California, Irvine, Irvine, USA
yanqig1@uci.edu, stasio@ics.uci.edu

² Algorand Foundation, New York, USA
hugo@algorand.foundation

Abstract. OPAQUE [Jarecki et al., Eurocrypt 2018] is an asymmetric password authenticated key exchange (aPAKE) protocol that is being developed as an Internet standard and for use within TLS 1.3. OPAQUE combines an Oblivious PRF (OPRF) with an authenticated key exchange to provide strong security properties, including security against pre-computation attacks (called saPAKE security). However, the security of OPAQUE relies crucially on the security of the OPRF. If the latter breaks (by cryptanalysis, quantum attacks or security compromise), the user's password is exposed to an offline dictionary attack. To address this weakness, we present KHAPE, a variant of OPAQUE that does not require the use of an OPRF to achieve aPAKE security, resulting in improved resilience and near-optimal computational performance. An OPRF can be *optionally* added to KHAPE, for enhanced saPAKE security, but without opening the password to an offline dictionary attack upon OPRF compromise.

In addition to resilience to OPRF compromise, a DH-based implementation of KHAPE (using HMQV) offers the best performance among aPAKE protocols in terms of exponentiations with less than the cost of an exponentiation on top of an UNauthenticated Diffie-Hellman exchange. KHAPE uses three messages if the server initiates the exchange or four when the client does (one more than OPAQUE in the latter case).

All results in the paper are proven within the UC framework in the ideal cipher model. Of independent interest is our treatment of *key-hiding AKE* which KHAPE uses as a main component as well as our UC proofs of AKE security for protocols 3DH (a basis of Signal), HMQV and SKEME, that we use as efficient instantiations of KHAPE.

1 Introduction

In the last few years the subject of password authenticated key exchange (PAKE) protocols, particularly in the client-server setting (called *asymmetric* PAKE, or aPAKE for short), has seen renewed interest due to the weaknesses of password protocols and the ongoing standardization effort at the Internet Engineering Task Force [49]. In particular, due to vulnerabilities in PKI systems and TLS deployment, the standard PKI-based encrypted password authentication

(or “password-over-TLS”) often leads to disclosure of passwords and increased exploitation of phishing techniques. Even when the password is decrypted at the correct server, its presence in plaintext form after decryption, constitutes a security vulnerability as evidenced by repeated incidents where plaintext passwords were accidentally stored in large quantities and for long periods of time even by security-conscious companies [1, 2].

In this paper we investigate the question of *how “minimal” an asymmetric PAKE can be*. In spite of the many subtleties surrounding the design and analysis of aPAKE protocols, there are several efficient and practical realizations which meet a universally composable (UC) notion of aPAKE [27]. For example, the overhead of the recently analyzed SPAKE2+ protocol [51] over the *unauthenticated* Diffie-Hellman (uDH) protocol is 1 or 2 exponentiations per party. Similar overhead costs are also imposed by the generic results which compile any PAKE to aPAKE [27, 33]. Known *strong* aPAKEs (see below), add similar or larger overhead costs [15, 37].

The comparison to uDH is significant not only from a practical point of view, but also because PAKE protocols imply unauthenticated key exchange in the sense of the Impagliazzo-Rudich results [29, 34]. Thus, we can see uDH as the lowest possible expected performance of PAKE protocols. But how close to the uDH cost can we get; can one improve on existing protocols?

In the symmetric PAKE case, where the two peers share the same password, there are almost optimal answers to this question. The Bellare-Meritt’s classical EKE protocol [10], shows that all you need is to apply a symmetric-key encryption on top of the uDH transcript. It requires a carefully chosen encryption scheme, e.g., one that is modeled after an ideal cipher, but it only involves symmetric key techniques [4, 9, 14, 46].¹

Can this low overhead relative to uDH be achieved also in the more involved setting of asymmetric PAKEs, where security against offline attacks is to be provided even when the server is broken into? We show an aPAKE protocol, KHAPE, that only requires symmetric operations (in the ideal cipher model) over regular authenticated DH.

KHAPE (for Key-Hiding Asymmetric Pake) can be seen as a variant of the OPAQUE protocol [37] that is being developed into an Internet standard [42] and intended for use within TLS 1.3 [52]. OPAQUE introduces the idea of password-encrypted credentials containing an encrypted private key for the user and an authenticated public key for the server. The user deposits the encrypted credentials at the server during password registration and it retrieves them for login sessions, thus allowing user and server to run a regular authenticated key exchange (AKE) protocol. However, encrypting and authenticating credentials with a password opens the protocol to trivial offline dictionary attacks. Therefore, OPAQUE first runs an Oblivious PRF (OPRF) on the user’s password in order to derive a strong encryption key for the credential. This makes the

¹ Several other symmetric PAKE protocols, e.g. SPAKE2 [5], SPEKE [30, 35, 43] and TBPEKE [48], attain universally composable security without relying on an ideal cipher but incur additional exponentiations over uDH costs [3].

protocol fully reliant on the strength of the OPRF. If OPRF is ever broken (by cryptanalysis, quantum attacks or security compromise), the user’s password is exposed to an offline dictionary attack.

Near-optimal aPAKE. KHAPE addresses this weakness by dispensing with the OPRF (hence also improving performance). It uses a “paradoxical” mechanism that allows to directly encrypt credentials with the password and still prevent dictionary attacks. Two key ideas are: (i) dispense with authentication of the credentials² and instead use a non-committing encryption where decryption of a given ciphertext under different keys cannot help identify which key from a candidate set was used to produce that ciphertext; and (ii) using a *key-hiding* AKE. The latter refers to AKE protocols that require that no adversary, not even active one, can identify the long-term keys used by the peers to an exchange even if provided with a list of candidate keys (a notion reminiscent of key anonymity for public key encryption [8]).

Fortunately, many established AKE protocols are key hiding, including implicitly authenticated protocols such as 3DH [44] and HMQV [41], and KEM-based protocols with key-hiding KEMs (e.g., SKEME [39]). The non-committing property of encryption models symmetric encryption as an ideal model (similarly to the case of EKE discussed above) and allows for implementations based on random oracles with hash-to-curve operations to encode group elements as strings (see Sect. 8). As a result, KHAPE with HMQV, uses only one fixed-base exponentiation, one variable-base (multi)exponentiation for each party, and one hash-to-curve operation for the client. In all, it achieves computational overhead relative to *unauthenticated* Diffie-Hellman of *less than the cost of one exponentiation*, thus providing a close-to-optimal answer to our motivating questions above. Such computational performance compares favorably to that of other efficient aPAKE protocols such as SPAKE2+ and OPAQUE that incur overhead of one and two (variable-base) exponentiations, respectively, for server and client. In terms of number of messages, KHAPE uses 4 (3 if server initiates), compared to 3 messages in SPAKE2+ and OPAQUE.

Refer to Sect. 6 for a detailed description and rationale of the generic KHAPE protocol (compiling any key-hiding AKE into an aPAKE) and to Sect. 7 for the instantiation using HMQV.

On Strong aPAKE and reliance on OPRF. In the comparisons above, it is important to stress that OPAQUE achieves a *stronger notion of aPAKE*, the so called Strong aPAKE (saPAKE) model from [37]. In this model, the attacker that compromises a server can only start running an offline dictionary attack after breaking into the server. In contrast, in regular aPAKE, an offline attack is still needed but a specialized dictionary can be prepared ahead of time and used to find the password almost instantaneously when breaking into the server. KHAPE, as discussed above, does not provide this stronger security. However, as shown in [37], one can add a run of an OPRF to any aPAKE protocol to achieve

² Dispensing with authentication of credentials in OPAQUE completely breaks the protocol, allowing for trivial offline dictionary attacks.

Strong aPAKE security. If one does that to KHAPE, one gets a Strong aPAKE protocol with performance similar to that of OPAQUE (using HMQV or 3DH).

However, there is a significant difference in the reliance on the security of OPRF. While the password security of OPAQUE breaks down with a compromise of the OPRF key (namely, it allows for an offline dictionary attack on the password), in KHAPE the effect of compromising the OPRF is only to fall back to the (non-strong) aPAKE setting. In particular, this distinction is relevant in the context of quantum-safe cryptography as there are currently no known efficient OPRFs considered to be quantum safe. This opens a path to quantum-safe aPAKEs based on KHAPE with key hiding quantum-safe KEMs.

Closer comparison with OPAQUE. As stated above, KHAPE has an advantage over OPAQUE in terms of security due to its weaker reliance on OPRF and its computational advantage when the OPRF is not used. Also, KHAPE seems more conducive to post-quantum security via post-quantum key-hiding KEMs.³ On the other hand, KHAPE requires one more message and allows for a more restrictive family of AKEs relative to OPAQUE (e.g., it does not allow for signature-based protocols as those based on SIGMA [40] and used in TLS 1.3 and IKEv2). KHAPE also relies for its analysis on the ideal cipher model while OPAQUE uses the random oracle model. An interesting advantage of KHAPE over OPAQUE is that in OPAQUE, an online attacker testing a password learns whether the password was wrong before the server does (in KHAPE the server learns first). This leads to a more complex mechanism for counting password failures at a server running OPAQUE, especially in settings with unreliable communication. Finally, we point out an advantage of using an OPRF with KHAPE (in addition to providing Strong aPAKE security): It allows for multi-server security via a threshold OPRF [36] where an attacker needs to break into multiple servers before it can run an offline attack on a password.

UC model analysis of (key-hiding) AKE's. All our protocols are framed and analyzed in the Universally Composable (UC) model [17]. This includes a formalization of the key-hiding AKE functionality that underlies the design of KHAPE. In order to instantiate KHAPE with specific AKE protocols, we prove that protocols 3DH [44] and HMQV [41] realize the key-hiding AKE functionality (in the ROM and under the Gap CDH assumption). We prove a similar result for SKEME [39] with appropriate KEM functions. We see the security analysis of these AKE protocols in the UC model, with and without key confirmation, as a contribution of independent interest. Moreover, the study of key-hiding AKE has applicability in other settings, e.g., where a gateway or IP address hides behind it other identities; say, a corporate site hosting employee identities or a web server aggregating different websites.

Organization. In Sect. 2, we define the notion of UC key-hiding AKE. In Sects. 3 and 4, we show, respectively, that 3DH and HMQV, are secure UC key-hiding AKE protocols under the Gap DH assumption in ROM. In Sect. 5, we study

³ We are currently investigating the use of NIST's post-quantum KEM selections [47] in conjunction with KHAPE.

the security of the SKEME protocol as a key-hiding AKE. In Sect. 6, we show a compiler from key-hiding AKE to asymmetric PAKE. In Sect. 7 we describe a concrete example of aPAKE, KHAPE-HMQV, that instantiates KHAPE with HMQV as the key-hiding AKE. Finally, in Sect. 8 we survey potential instantiations of our ideal cipher encryption. In the full version of the paper [28], we include more background material as well as full proofs for all our theorems.

2 The Key-Hiding AKE UC Functionality

Protocol KHAPE results from the composition of an encrypted credentials scheme and a *key-hiding* AKE protocol. Figure 1 defines the UC functionality $\mathcal{F}_{\text{khAKE}}$ that captures the properties required from a key-hiding AKE protocol. The modeling choices target the following requirements: First, as shown in Sect. 6, the security and key-hiding properties of this key-hiding AKE model suffice for our main application, a generic construction of UC aPAKE from any protocol realizing $\mathcal{F}_{\text{khAKE}}$. Second, as we show in the final version [28], adding a standard key confirmation to any protocol that realizes $\mathcal{F}_{\text{khAKE}}$ results in a (standard) UC AKE with explicit entity authentication. Lastly, this functionality is realized by several well-known and efficient AKE protocols, including 3DH and HMQV, as shown in Sects. 3 and 4, as well as by a KEM-based AKE such as SKEME, if instantiated with a key-hiding KEM, see Sect. 5. We provide more details and rationale for the $\mathcal{F}_{\text{khAKE}}$ next.

High-level requirements for key-hiding AKE. The most salient property we require from AKE is *key hiding*. To illustrate this requirement consider an experiment where the attacker \mathcal{A} is provided with a transcript of a session between a party P and its counterparty CP . Party P has two inputs in this AKE instance: a public key pk_{CP} for CP and its own private key sk_P which P uses to authenticate to CP who presumably knows P 's public key pk_P . In addition, \mathcal{A} is given a pair of private keys: P 's private key sk_P and a second random independent private key. \mathcal{A} 's goal is to decide which of the two keys P used in that session.⁴ We are interested in AKE protocols where the attacker has no better chance to answer correctly than guessing randomly *even for sessions in which \mathcal{A} is allowed to choose the messages from CP .*

The key hiding property will come up in the analysis of KHAPE as follows. The attacker learns a ciphertext c that encrypts the user's private key under the user's password. By decrypting this ciphertext under all passwords in a dictionary, the attacker obtains a set of possible private keys for the user. The key hiding property ensures that the attacker cannot identify the correct key (or the password) in the set. Fortunately, as we prove here, a large class of AKE protocols satisfy the key-hiding property, including implicitly authenticated protocols such as HMQV and 3DH, and some KEM-based protocols.

⁴ This is reminiscent of key anonymity for PK encryption [8] where the attacker needs to distinguish between public keys for a given ciphertext.

- PK is the list of all public keys created via `Init`, initially empty
- PK_P is the list of all public keys created by P , initially empty for all P
- CPK is the list of all compromised keys in PK , initially empty

Keys: Initialization and Attacks

On `Init` from P :

Send `(Init, P)` to \mathcal{A} , let \mathcal{A} specify pk s.t. $pk \notin PK$, add pk to PK and to PK_P , and output `(Init, pk)` to P

On `(Compromise, P, pk)` from \mathcal{A} :

If $pk \in PK_P$ then add pk to CPK

Login Sessions: Initialization and Attacks

On `(NewSession, sid, CP, pk, pkCP)` from P :

If $pk \in PK_P$ and there is no prior session record $\langle \text{sid}, P, \cdot, \cdot, \cdot, \cdot \rangle$ then:

- create session record $\langle \text{sid}, P, CP, pk, pk_{CP}, \perp \rangle$ marked fresh
- initialize random function $R_P^{\text{sid}} : (\{0, 1\}^*)^3 \rightarrow \{0, 1\}^\kappa$
- send `(NewSession, sid, P, CP)` to \mathcal{A}

On `(Interfere, sid, P)` from \mathcal{A} :

If session $\langle \text{sid}, P, CP, pk_P, pk_{CP}, \perp \rangle$ is marked fresh then change its mark to interfered

Login Sessions: Key Establishment

On `(NewKey, sid, P, α)` from \mathcal{A} :

If \exists session record $\text{rec} = \langle \text{sid}, P, CP, pk_P, pk_{CP}, \perp \rangle$ then:

- if rec is marked fresh: If \exists record $\langle \text{sid}, CP, P, pk_{CP}, pk_P, k' \rangle$ marked fresh s.t. $k' \neq \perp$ then set $k \leftarrow k'$, else pick $k \leftarrow_{\mathcal{R}} \{0, 1\}^\kappa$
- if rec is marked interfered then set $k \leftarrow R_P^{\text{sid}}(pk_P, pk_{CP}, \alpha)$
- update rec to $\langle \text{sid}, P, CP, pk_P, pk_{CP}, k \rangle$ and output `(NewKey, sid, k)` to P

Session-Key Query

On `(SessionKey, sid, P, pk, pk', α)` from \mathcal{A} :

If \exists record $\langle \text{sid}, P, \dots \rangle$ and $pk' \in CPK$ or $pk' \notin PK$, send $R_P^{\text{sid}}(pk, pk', \alpha)$ to \mathcal{A}

Fig. 1. $\mathcal{F}_{\text{khAKE}}$: functionality for key-hiding AKE

Additionally, $\mathcal{F}_{\text{khAKE}}$ strengthens the basic guarantees of AKE protocols in several ways. It requires resilience to KCI (key-compromise impersonation) attacks, namely, upon the compromise of the private key of party P , the attacker can impersonate P to others but it cannot impersonate others to P . In the aPAKE setting, this ensures that an attacker that compromises a server, cannot impersonate the client to the server without going through an offline dictionary attack. In the context of key hiding AKE, we also need KCI resilience to prevent the attacker from authenticating to the client when given a set of possible private keys for that client.

Second, $\mathcal{F}_{\text{khAKE}}$ requires that keys exchanged by a honest P with a corrupted CP still maintain a good amount of randomness, namely, the attacker can cause them to deviate from uniform but not by much (a property sometimes referred to as “contributive” key exchange, and not required in standard UC treatment). In the setting of protocol KHAPE, adversarial choice of session keys (particularly the ability of the attacker to create equal keys in different sessions) could lead to protocols where the attacker can test more than one password in a single session.

Properties that we do *not* consider as part of the $\mathcal{F}_{\text{khAKE}}$ functionality, but will be provided by our final aPAKE protocol, KHAPE, include key confirmation, explicit authentication and full forward secrecy ($\mathcal{F}_{\text{khAKE}}$ itself implies forward secrecy only against passive attackers).

Identities and public keys. We consider a setting where each party P has multiple public keys in the form of arbitrary handles pk . In the security model we assume that the public keys are arbitrary bitstrings chosen without loss of generality by the attacker (ideal adversary) \mathcal{A} , with the limitation that honest parties are assigned non-repeating pk strings. Pairs (P, pk) act as regular UC identities from the environment’s point of view, but the pk component is concealed from \mathcal{A} during key exchange sessions, even for sessions which are actively attacked by \mathcal{A} . This model can capture practical settings where P represents a gateway or IP address behind which other identities reside, e.g., a corporate site hosting employee identities or a web server aggregating different websites, and where one is interested to hide which party behind the gateway is communicating in a given session. Our specific application setting when using key-hiding AKE in the aPAKE construction of Sect. 6, is more abstract: The party symbols P, CP represent parties like internet clients and servers, while the multiplicity of public keys comes from decryptions of encrypted credentials under multiple password.

(Compromise, P, pk). This adversarial action hands the (long-term) private key of party (P, pk) to the attacker \mathcal{A} . Such private-key leakage does not provide \mathcal{A} with control over party P , and it does not even imply that the sessions which party P runs using the (leaked) key pk are insecure. However, when combined with the ability to run active attacks, via the *Interfere* action below, \mathcal{A} can fully impersonate (P, pk) in sessions of \mathcal{A} ’s choice. The leakage of the private key sk corresponding to (P, pk) does not affect the security of a session executed by party P even if it uses the compromised key pk . This captures the KCI property, i.e. that leakage of the private key of party P does not allow to impersonate others to party P . Also, any party P' which runs AKE with a *counterparty* identity specified as (P, pk) , will also be secure as long as \mathcal{A} does not actively interfere in that protocol. This captures the requirement that passively-observed AKE instance are secure regardless of the compromise of the long-term secrets used by either party. Note that \mathcal{A} cannot compromise a party P but rather an identity pair (P, pk) and such compromise does not affect other pairs (P, pk') .

NewSession. A session is initiated by a party P that specifies its own identity pair (P, pk) as well as the intended counterparty identity pair (CP, pk_{CP}) . Session identifiers sid are assumed to be unique within an honest party. The role of the initialized session-specific random function R_P^{sid} is described below. A record for a session is initialized as *fresh* and is represented by a tuple $(sid, P, CP, pk, pk_{CP}, \perp)$ where the last position, set to \perp , is reserved for recording the session key. An *essential element* in **NewSession** is that \mathcal{A} learns (sid, P, CP) but *it does not learn* (pk, pk_{CP}) . In the real world this translates into the inability of the attacker to identify public (or private) keys associated to a pair of parties (P, CP) engaging in the Key-Hiding AKE protocol.

The functionality enforces that an honest P can start a session only on key pk which P generated and for which it holds a private key. However, the functionality does not check anything about the intended counterparty's identity (CP, pk_{CP}) , so the private key corresponding to pk_{CP} could be held by party CP , or it could be held by a different party, or it could be compromised by the adversary, or it could be that pk_{CP} was not even generated by the key generation interface of \mathcal{F}_{khAKE} , and it is an *adversarial* public key, whose private key the environment gave to the adversary. Our model thus includes honest parties who are tricked to use a wrong public key for the counterparty (e.g., via a phishing attack) in which case the attacker may know the corresponding private key. Note that regardless of what key pk_{CP} the session runs on, it is not given to the adversary, so if it is a key created by the environment (i.e. a higher-level application which uses the key-hiding AKE) it does not necessarily follow that this key will be known to the adversary, and only in the case it is known the adversary will be able to attack that session using interfaces **Interfere**, **NewKey**, and **SessionKey** below.

Function R_P^{sid} . When command **NewSession** creates a session for (sid, P) the functionality initializes a *random* function R_P^{sid} specific to this session. Function R_P^{sid} is used to set the value of the session key for sessions in which \mathcal{A} actively interferes. It also allows \mathcal{A} to have limited control over the value of the key under strict circumstances, namely it must know the public keys pk, pk_{CP} used on that session, and it must compromise party (CP, pk_{CP}) . Even then the only freedom \mathcal{A} has is to evaluate function R_P^{sid} on any point α via a **SessionKey** query, see below, and then choose one such point in the **NewKey** command. This captures the “contributive” property discussed above: If an honest party runs the AKE protocol even with adversary as a counterparty, the adversary's influence over the session key is limited to pre-computing polynomially-many random key candidates and then choosing one of them as a key on that session. The exact mechanics and functionality of R_P^{sid} are defined in the **NewKey** and **SessionKey** actions below.

(Interfere, sid, P). This action represents an active attack on session (P, sid) and makes the session change its status from *fresh* to *interfered*. The adversary does not have to know either P 's own key pk or the intended counterparty key pk_{CP}

which P uses on that session.⁵ Such active attack will prevent session (P, sid) from establishing a secure key with any other honest party session, e.g. (CP, sid) . It will also allow \mathcal{A} to learn and/or influence the value of the session key this session outputs (using function R_P^{sid}), but only if in addition to being active \mathcal{A} compromises the counterparty key (CP, pk_{CP}) used on session (P, sid) .

NewKey. This action finalizes an AKE instance and makes (P, sid) output a session key. If the session is **fresh** then it receives either a fresh random key or the same key that was previously received by a matching session. If the session is **interfered**, the value of the session key is determined by the function R_P^{sid} on input (pk, pk_{CP}, α) where α is chosen arbitrarily by \mathcal{A} , allowing \mathcal{A} to influence the value of the session key (but in a very limited way as explained above). In the real-world, α represents transcript elements generated by the attacker, e.g., value Y an adversarial P_2 sends to an honest party P_1 in 3DH or HMQV.

SessionKey. This action allows \mathcal{A} to query the function R_P^{sid} associated to a session (sid, P) , potentially allowing \mathcal{A} to learn and/or influence the session key for (sid, P) . Note that learning any values of function R_P^{sid} is useless unless the adversary actively attacks session (sid, P) , because otherwise R_P^{sid} is not used to determine the key output by session (sid, P) . Moreover, \mathcal{A} needs to provide (pk, pk_{CP}, α) as input to **SessionKey**, and if those inputs do not match P 's own key pk and the intended counterparty key pk_{CP} which P uses on session (sid, P) , then this query reveals an irrelevant value, since R_P^{sid} is a random function. Finally, $\mathcal{F}_{\text{khAKE}}$ releases value $R_P^{\text{sid}}(pk, pk_{CP}, \alpha)$ to \mathcal{A} only if key pk_{CP} is either compromised or adversarial. Summing up, the ability to learn (and/or control via the **NewKey** interface) the session key output by session (sid, P) is restricted to the case where all of the following hold: \mathcal{A} actively interfered on that session, \mathcal{A} guesses keys pk, pk_{CP} which this session uses, and \mathcal{A} compromises counterparty's key (CP, pk_{CP}) .

How $\mathcal{F}_{\text{khAKE}}$ ensures key hiding and session security. The description of $\mathcal{F}_{\text{khAKE}}$ is now complete. We now explain how $\mathcal{F}_{\text{khAKE}}$ ensures the key hiding property by which \mathcal{A} cannot learn the value pk for an identity pair (P, pk) even if \mathcal{A} knows P , has a list of all possible values of (P, pk) , and actively interacts with (P, pk) using a compromised party (CP, pk_{CP}) . Let's assume these conditions hold. Note that the only actions in which \mathcal{A} can learn pk values from $\mathcal{F}_{\text{khAKE}}$ are upon key generation and via the **SessionKey** call. Key generation assumes that \mathcal{A} has a list of all possible values (P, pk) . As we explain above, the only argument on which the value of function R_P^{sid} is useful is a tuple (pk, pk_{CP}, α) which the functionality uses to derive a session key for an actively attacked session (sid, P) .

Consequently, the only way $\mathcal{F}_{\text{khAKE}}$ can leak the session key output by (sid, P) is if \mathcal{A} satisfies the three conditions above, i.e. it interferes in that session, key

⁵ Currently functionality $\mathcal{F}_{\text{khAKE}}$ assumes the ideal-world adversary \mathcal{A} knows, and indeed creates, all honest parties' public keys. A tighter model is possible, if $\mathcal{F}_{\text{khAKE}}$ samples public keys on behalf of honest players using the prescribed key generation algorithm, instead of letting \mathcal{A} pick them. This would allow modeling use cases where the public keys are not public and are not freely available to the adversary.

pk_{CP} used on that session is either compromised or adversarial, and \mathcal{A} queries `SessionKey` on the proper keys pk, pk_{CP} . This is also the only way \mathcal{A} can learn anything about keys pk, pk_{CP} used by session (sid, P) : It has to attack the session, compromise pk_{CP} , get a session key candidate k^* via query `SessionKey` on pk, pk_{CP} , and then compare this key candidate against any information it has about the key k output by session (sid, P) . For example, if P 's higher-level application uses key k to MAC or encrypt a message, the adversary can verify the result against a candidate key k^* and thus learn whether $k^* = k$, and hence whether keys pk, pk_{CP} which \mathcal{A} used to compute k^* were the same keys that were used by session (sid, P) .

3 3DH as Key-Hiding AKE

We show that protocol 3DH, presented in Fig. 2, realizes the UC notion of Key-Hiding AKE, as defined by functionality $\mathcal{F}_{\text{khAKE}}$ in Sect. 2, under the Gap CDH assumption in ROM. As a consequence, 3DH can be used to instantiate protocol KHAPE in a simple and efficient way.

3DH [44] is a simple, implicitly authenticated key exchange used as the basis of the X3DH protocol [45] that underlies the Signal protocol. It consists of a plain Diffie-Hellman exchange authenticated via the session-key derivation that combines the ephemeral and long-term key of both peers. Specifically, if (a, A) and (b, B) are the long-term key pairs of two parties P_1 and P_2 , and (x, X) and (y, Y) are their ephemeral DH values, then 3DH combines these key pairs to compute a (hash of) the *triple* of Diffie-Hellman values, $\sigma = g^{xb} \| g^{ay} \| g^{xy}$. Security of 3DH is intuitively easy to see: It follows from the fact that to compute σ the attacker must either (1) know (x, a) to attack party P_2 who uses A as a public key for its counterparty, or (2) know (y, b) to attack party P_1 who uses B as a public key for its counterparty. In other words, the attacker wins only if it is an active man-in-the-middle attacker *and* it compromises the key used as counterparty's public key by the attacked party. (Recall that "compromising a public key" stands for learning the corresponding private key.) The key-hiding property comes from the fact that the values X and Y exchanged in the protocol do not depend on long-term keys, and the fact that the only information about the long-term keys used by any party can be gleaned only from the session key they output and from H oracle queries on a σ value computed using these keys. The formal proof of key-hiding in the UC model captures this argument, and we present it below.

We note that 3DH is not the most efficient key-hiding AKE. 3DH costs one fixed-base and three variable-base exponentiations per party, and in Sect. 4 we will show that HMQV, which preserves the bandwidth and round complexity of 3DH but folds the three variable-base exponentiations of 3DH into a single multi-exponentiation, realizes the key-hiding AKE functionality under the same Gap CDH assumption (although with worse exact security guarantees). However, HMQV can be seen as a modification of 3DH, and the security analysis of 3DH we show below will form a blueprint for the analysis of HMQV in Sect. 4.

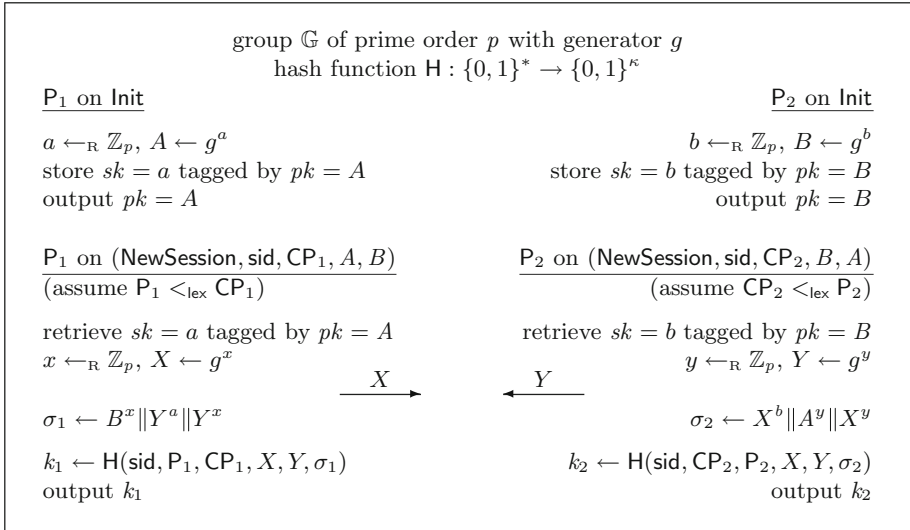


Fig. 2. Protocol 3DH: “Triple Diffie-Hellman” key exchange

Conventions.

(1) In Fig. 2 we assume that each party runs 3DH using key pair (sk, pk) previously generated via procedure Init. In Fig. 2 these are resp. (a, A) for P_1 and (b, B) for P_2 . Note that no such requirement is posed on the counterparty public key each party uses, resp. public key B used by P_1 and A used by P_2 .

(2) We implicitly assume that each party P_i uses its own identity as a protocol input, together with the identity CP_i of its assumed counterparty. These identities could be e.g. domain names, user names, or any other identifiers. They have no other semantics except that the two parties can establish the same session key only if they assume matching identifiers, i.e. $(P_1, CP_1) = (CP_2, P_2)$.

(3) Protocol 3DH is symmetric except for the ordering of group elements in tuple σ and the ordering of elements in the inputs to hash H . Each protocol party P can locally determine this order based on whether string P is lexicographically smaller than string CP . (In Fig. 2 we assume that $P_1 <_{\text{lex}} P_2$.) An equivalent way to see it is that each party P computes a “role” bit $\text{role} \in \{1, 2\}$ and follows the protocol of party P_{role} in Fig. 2: Party P sets this bit as $\text{role} = 1$, called the “client role”, if $P <_{\text{lex}} CP$, and $\text{role} = 2$, called the “server role”, otherwise.

(4) We assume that parties verify public keys and ephemeral DH values, resp. B, Y for P_1 and A, X for P_2 , as group \mathbb{G} elements. Optionally, instead of group membership testing one can use cofactor exponentiation to compute σ .

Cryptographic Setting: Gap CDH and RO Hash. Let g generate a cyclic group \mathbb{G} of prime order p . The Computational Diffie-Hellman (CDH) assumption on \mathbb{G} states that given $(X, Y) = (g^x, g^y)$ for $(x, y) \leftarrow_{\mathbb{R}} (\mathbb{Z}_p)^2$ it is hard to find $\text{cdh}_g(X, Y) = g^{xy}$. The Gap CDH assumption states that CDH is hard even if the adversary has access to a Decisional Diffie-Hellman oracle ddh_g , which on input (A, B, C) returns 1 if $C = \text{cdh}_g(A, B)$ and 0 otherwise.

Theorem 1. *Protocol 3DH shown in Fig. 2 realizes $\mathcal{F}_{\text{khAKE}}$ if the Gap CDH assumption holds and \mathbf{H} is a random oracle.*

Initialization: Initialize an empty list $KL_{\mathcal{P}}$ for each \mathcal{P}

On (Init, \mathcal{P}) from \mathcal{F} :
 pick $sk \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, set $pk \leftarrow g^{sk}$, add (sk, pk) to $KL_{\mathcal{P}}$, and send pk to \mathcal{F}

On \mathcal{Z} 's permission to send (Compromise, \mathcal{P}, pk) to \mathcal{F} :
 if $\exists (sk, pk) \in KL_{\mathcal{P}}$ send sk to \mathcal{A} and (Compromise, \mathcal{P}, pk) to \mathcal{F}

On (NewSession, sid, \mathcal{P}, CP) from \mathcal{F} :
 if $\mathcal{P} <_{\text{lex}} \text{CP}$ then set $\text{role} \leftarrow 1$ else set $\text{role} \leftarrow 2$
 pick $w \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, store $\langle \text{sid}, \mathcal{P}, \text{CP}, \text{role}, w \rangle$, send $W = g^w$ to \mathcal{A}

On \mathcal{A} 's message Z to session \mathcal{P}^{sid} (only first such message counts):
 if \exists record $\langle \text{sid}, \mathcal{P}, \text{CP}, \cdot, w \rangle$:
 if \exists no record $\langle \text{sid}, \text{CP}, \mathcal{P}, \cdot, z \rangle$ s.t. $g^z = Z$ then send (Interfere, sid, \mathcal{P}) to \mathcal{F}
 send (NewKey, sid, \mathcal{P}, Z) to \mathcal{F}

On query $(\text{sid}, \text{C}, \text{S}, X, Y, \sigma)$ to random oracle \mathbf{H} :
 if $\exists \langle (\text{sid}, \text{C}, \text{S}, X, Y, \sigma), k \rangle$ in $\mathbf{T}_{\mathbf{H}}$ then output k , else pick $k \leftarrow_{\mathbb{R}} \{0, 1\}^k$ and:
 if \exists record $\langle \text{sid}, \text{C}, \text{S}, 1, x \rangle$ and $(a, A) \in KL_{\mathcal{C}}$ s.t. $(X, \sigma) = (g^x, (B^x \| Y^a \| Y^x))$
 for some B , send (SessionKey, sid, C, A, B, Y) to \mathcal{F} , if \mathcal{F} returns k^* reset $k \leftarrow k^*$
 if \exists record $\langle \text{sid}, \text{S}, \text{C}, 2, y \rangle$ and $(b, B) \in KL_{\mathcal{S}}$ s.t. $(Y, \sigma) = (g^y, (X^b \| A^y \| X^y))$
 for some A , send (SessionKey, sid, S, B, A, X) to \mathcal{F} , if \mathcal{F} returns k^* reset $k \leftarrow k^*$
 add $\langle (\text{sid}, \text{C}, \text{S}, X, Y, \sigma), k \rangle$ to $\mathbf{T}_{\mathbf{H}}$ and output k

Fig. 3. Simulator SIM showing that 3DH realizes $\mathcal{F}_{\text{khAKE}}$ (abbreviated “ \mathcal{F} ”)

Proof Overview. We show that for any efficient environment algorithm \mathcal{Z} , its view of the *real-world* security game, i.e. an interaction between the real-world adversary and honest parties who follow protocol 3DH, is indistinguishable from its view of the *ideal-world* game, i.e. an interaction between the ideal-world adversary, whose role is played by the *simulator*, with the functionality $\mathcal{F}_{\text{khAKE}}$. We show the simulator algorithm SIM in Fig. 3. The real-world game, Game 0, is shown in Fig. 4, and the ideal-world game defined by a composition of algorithm SIM and functionality $\mathcal{F}_{\text{khAKE}}$, denoted Game 7, is shown in Fig. 5.

Initialization: Initialize an empty list KL_P for each P

On message Init to P :
 pick $sk \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, set $pk \leftarrow g^{sk}$, add (sk, pk) to KL_P , and output (Init, pk)

On message $(\text{Compromise}, P, pk)$:
 If $\exists (sk, pk) \in KL_P$ then output sk

On message $(\text{NewSession}, \text{sid}, CP, pk_P, pk_{CP})$ to P :
 if $\exists (sk, pk_P) \in KL_P$, pick $w \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, write $\langle \text{sid}, P, CP, sk, pk_{CP}, w \rangle$, output $W = g^w$

On message Z to session P^{sid} (only first such message is processed):
 if \exists record $\langle \text{sid}, P, CP, sk_P, pk_{CP}, w \rangle$, set $\sigma \leftarrow ((pk_{CP})^w \| Z^{sk_P} \| Z^w)$,
 $k \leftarrow H(\text{sid}, \{P, CP, W, Z, \sigma\}_{\text{ord}})$, output $(\text{NewKey}, \text{sid}, k)$

On H query $(\text{sid}, C, S, X, Y, \sigma)$:
 if $\exists \langle (\text{sid}, C, S, X, Y, \sigma), k \rangle$ in \overline{T}_H then output k , else pick $k \leftarrow_{\mathbb{R}} \{0, 1\}^\kappa$ and:
 add $\langle (\text{sid}, C, S, X, Y, \sigma), k \rangle$ to T_H and output k

Fig. 4. 3DH: Environment’s view of real-world interaction (Game 0)

As is standard, we assume that the real-world adversary \mathcal{A} is a subroutine of the environment \mathcal{Z} , therefore the sole party that interacts with Games 0 or 7 is \mathcal{Z} , issuing commands Init and NewSession to honest parties P , adaptively compromising public keys, and using \mathcal{A} to send protocol messages Z to honest party’s sessions and making hash function H queries. The proof follows a standard strategy of showing a sequence of games that bridge between Game 0 and Game 7, where at each transition we argue that the change is indistinguishable. We use G_i to denote the event that \mathcal{Z} outputs 1 while interacting with Game i , and the theorem follows if we show that $|\Pr[G_0] - \Pr[G_7]|$ is negligible under the stated assumptions.

Notation. To make the real-world interaction in Fig. 4 more concise, we adopt a notation which stresses the symmetric nature of 3DH protocol: We use variable $W = g^w$ to denote the message which party P sends out, and variable Z to denote the message it receives, e.g. $(W, Z) = (X, Y)$ if P plays the “client” role and $(W, Z) = (Y, X)$ if P plays the “server” role. If $\sigma = \sigma_1 \| \sigma_2 \| \sigma_3$ then let $\{\sigma\}_{\text{flip}} = \sigma_2 \| \sigma_1 \| \sigma_3$. We will use $\{P, CP, W, Z, \sigma\}_{\text{ord}}$ to denote string P, CP, W, Z, σ if $P <_{\text{lex}} CP$ or string $CP, P, Z, W, \{\sigma\}_{\text{flip}}$ if $CP <_{\text{lex}} P$. With this notation each party’s 3DH protocol code can be restated in the symmetric way, as in Fig. 4, because session key computation of party P can be denoted in a uniform way as $k \leftarrow H(\text{sid}, \{P, CP, W, Z, \sigma\}_{\text{ord}})$ for $\sigma = (pk_{CP})^w \| Z^{sk_P} \| Z^w$.

We use the same symmetric notation to describe simulator SIM in Fig. 3 and the ideal-world game implied by SIM and $\mathcal{F}_{\text{khAKE}}$ in Fig. 5, except for the way SIM treats H oracle queries, which we separate into two cases based on the roles played by the two parties whose sessions are potentially involved in any H query. In H-handling code of SIM we denote the identifiers of the two parties involved in a query as C and S, for the parties playing respectively the client and server roles, and the code that follows uses role-specific notation to handle attacks on the sessions executed respectively by C and S.

Throughout the proof we use P^{sid} to denote a session of party P with identifier sid. We use v_P^{sid} to denote a local variable v pertaining to session P^{sid} or a message v which this session receives, and whenever identifier sid is clear from the context we write v_P instead of v_P^{sid} . Note that session CP^{sid} is uniquely defined for every session P^{sid} by setting $CP = CP_P^{\text{sid}}$, and we will implicitly assume below that a counterparty's session is defined in this way.

For a fixed environment \mathcal{Z} , let q_K and q_{ses} be (the upper-bounds on) the number of resp. keys and sessions initialized by \mathcal{Z} , let q_H be the number of H oracle queries \mathcal{Z} makes, and let $\epsilon_{g\text{-cdh}}^{\mathcal{Z}}$ be the maximum advantage in solving Gap CDH in \mathbb{G} of an algorithm that makes q_H DDH oracle queries and uses the resources of \mathcal{Z} plus $O(q_H + q_{\text{ses}})$ exponentiations in \mathbb{G} .

Define the following two functions for every session P^{sid} :

$$3DH_P^{\text{sid}}(pk, pk', Z) = \text{cdh}_g(W, pk') \parallel \text{cdh}_g(pk, Z) \parallel \text{cdh}_g(W, Z) \text{ for } W = W_P^{\text{sid}} \quad (1)$$

$$R_P^{\text{sid}}(pk, pk', Z) = H(\text{sid}, \{P, CP_P^{\text{sid}}, W_P^{\text{sid}}, Z, 3DH_P^{\text{sid}}(pk, pk', Z)\}_{\text{ord}}) \quad (2)$$

If session P^{sid} runs on its own private key sk_P , counterparty's public key pk_{CP} , and receives message Z , then its output session key is $k = R_P^{\text{sid}}(pk_P, pk_{CP}, Z)$ for $pk_P = g^{sk_P}$. Note also that an adversary can locally compute function R_P^{sid} for any pk_P , any key pk_{CP} which was either generated by the adversary or it was generated by an honest party but it has been compromised, and any Z which the adversary generates, because the adversary can then compute functions $\text{cdh}_g(\cdot, pk_{CP})$ and $\text{cdh}_g(\cdot, Z)$ on any inputs.

Simulator. Simulator SIM, shown in Fig. 3, picks all (sk, pk) pairs on behalf of honest players and surrenders the corresponding private key whenever an honestly-generated public key is compromised. To simulate honest party P behavior the simulator sends $W = g^w$ for random w . When P^{sid} receives Z the simulator forks: If Z originated from honest session CP^{sid} which runs on matching identifiers (sid, CP, P) , SIM treats this as a case of honest-but-curious attack that connects two potentially matching sessions and sends NewKey to $\mathcal{F}_{\text{khAKE}}$. (Z included in this call is ignored by $\mathcal{F}_{\text{khAKE}}$.) Otherwise SIM treats it as an active attack on P^{sid} and sends Interfere followed by $(\text{NewKey}, \dots, Z)$. Note that in response $\mathcal{F}_{\text{khAKE}}$ will treat P^{sid} as interfered and set its output key as $k \leftarrow R_P^{\text{sid}}(pk_P, pk_{CP}, Z)$ where (pk_P, pk_{CP}) are the (own, counterparty) pair of public keys which P^{sid} uses, and which is unknown to SIM (except if pk_{CP} was generated by the adversary, in which case it was leaked to SIM at NewSession). Finally, SIM services H oracle queries $(\text{sid}, C, S, X, Y, \sigma)$ by identifying those that

pertain to viable session-key computations by either session C^{sid} or S^{sid} . We describe it here only for C^{sid} -side H queries since S^{sid} -side queries are handled symmetrically. If H query involves $\sigma = 3\text{DH}_C^{\text{sid}}(A, B, Y)$ for some A, B s.t. (1) A is one of the public keys generated by C , and (2) B is either some compromised honestly generated public key or it is an adversarial key which C^{sid} uses for the counterparty (recall that if C^{sid} runs on an adversary-generated counterparty key pk_{CP} then functionality $\mathcal{F}_{\text{khAKE}}$ leaks it to the adversary), then SIM treats that query as a potential computation of a session key output by C^{sid} , queries $(\text{SessionKey}, \text{sid}, C, A, B, Y)$ to $\mathcal{F}_{\text{khAKE}}$. If B is compromised or adversarial then $\mathcal{F}_{\text{khAKE}}$ responds with $k^* \leftarrow R_C^{\text{sid}}(A, B, Y)$ and SIM embeds k^* into H output. Note that if (A, B) matches the (own, counterparty) keys used by C^{sid} , and C^{sid} receives $Z = Y$ in the protocol, then k^* will match the session key output by C^{sid} . For all other triples (A, B, Y) the outputs of R_C^{sid} are irrelevant except that (1) if the adversary learns the real session key output by C^{sid} then these H outputs inform the adversary that pair (A, B) is *not* the (own, counterparty) key pair used by C^{sid} , and (2) if the adversary bets on some (A, B) pair used by C^{sid} then it can use H queries to find an “optimal” protocol response Y to C^{sid} for which the resulting (randomly sampled) session key has some properties the adversary likes, e.g. its last 20 bits are all zeroes, etc.

Game Sequence from Game 0 to Game 7. The full proof comprising the transitions between these games is presented in the full version [28].

4 HMQV as Key-Hiding AKE

We show that protocol HMQV [41], presented in Fig. 6, realizes the UC notion of Key-Hiding AKE, as defined by functionality $\mathcal{F}_{\text{khAKE}}$ in Sect. 2, under the Gap CDH assumption in ROM. It allows us to use HMQV with KHAPE, resulting in its most efficient instantiation, and, to the best of our knowledge the most efficient aPAKE protocol proposed. HMQV has been analyzed in [41] under the game-based AKE model of Canetti and Krawczyk [18], but the analysis we present is the first, to the best of our knowledge, to be done in the UC model.⁶

The logic of why HMQV is key hiding is similar to the case of 3DH. Namely, the only way to attack the privacy of party P which runs HMQV on inputs $(sk, pk) = (a, B)$, is to compromise the private key b corresponding to the public key B . (And symmetrically for the party that runs on $(sk, pk) = (b, A)$.) The HMQV equation, just like the 3DH key equation, involves both the ephemeral sessions secrets (x, y) and the long-term keys (a, b) , combining them in a DH-like formula $\sigma = g^{(x+da) \cdot (y+eb)}$ where d, e are hashes of (session state identifiers and) resp. $X = g^x$ and $Y = g^y$. Following essentially the same arithmetics as in the proof due to [41] shows that the only way to compute σ is to know either *both* x, a or *both* y, b , which means that the attacker must be (1) active, to chose the ephemeral session state variable resp. x or y , and (2) it must know the counterparty private key, resp. a or b .

⁶ However, we do not include adaptive session state compromise considered in [18, 41].

Initialization: Initialize empty lists: PK , CPK , and KL_P for all P

On message Init to P :
 set $sk \leftarrow_R \mathbb{Z}_p$, $pk \leftarrow g^{sk}$, send (Init, pk) to P , add pk to PK and (sk, pk) to KL_P

On message $(\text{Compromise}, P, pk)$:
 If $\exists (sk, pk) \in KL_P$ add pk to CPK and output sk

On message $(\text{NewSession}, \text{sid}, CP, pk_P, pk_{CP})$ to P :
 if $\exists (sk, pk_P) \in KL_P$ then:
 initialize random function $R_P^{\text{sid}} : (\{0, 1\}^*)^3 \rightarrow \{0, 1\}^\kappa$
 if $P <_{\text{lex}} CP$ then set $\text{role} \leftarrow 1$ else set $\text{role} \leftarrow 2$
 pick $w \leftarrow_R \mathbb{Z}_p$, write $\langle \text{sid}, P, CP, pk_P, pk_{CP}, \text{role}, w, \perp \rangle$ as fresh, output $W = g^w$

On message Z to session P^{sid} (only first such message is processed):
 if \exists record $\text{rec} = \langle \text{sid}, P, CP, pk_P, pk_{CP}, \text{role}, w, \perp \rangle$:
 if \exists record $\text{rec}' = \langle \text{sid}, CP, P, pk'_{CP}, pk'_P, \text{role}', z, k' \rangle$ s.t. $g^z = Z$
 then if rec' is fresh, $(pk_P, pk_{CP}) = (pk'_P, pk'_{CP})$, and $k' \neq \perp$:
 then $k \leftarrow k'$
 else $k \leftarrow_R \{0, 1\}^\kappa$
 else set $k \leftarrow R_P^{\text{sid}}(pk_P, pk_{CP}, Z)$ and re-label rec as interfered
 update rec to $\langle \text{sid}, P, CP, pk_P, pk_{CP}, \text{role}, w, k \rangle$, send $(\text{NewKey}, \text{sid}, k)$ to P

On H query $(\text{sid}, C, S, X, Y, \sigma)$:
 if $\exists \langle (\text{sid}, C, S, X, Y, \sigma), k \rangle$ in T_H then output k , else pick $k \leftarrow_R \{0, 1\}^\kappa$ and:
 1. if \exists record $\langle \text{sid}, C, S, \cdot, \cdot, 1, x, \cdot \rangle$ s.t. $(X, \sigma) = (g^x, (B^x \| Y^a \| Y^x))$ for some $(a, A) \in KL_C$ and B s.t. $B \in CPK$ or $B \notin PK$ then reset $k \leftarrow R_C^{\text{sid}}(A, B, Y)$
 2. if \exists record $\langle \text{sid}, S, C, \cdot, \cdot, 2, y, \cdot \rangle$ s.t. $(Y, \sigma) = (g^y, (X^b \| A^y \| X^y))$ for some $(b, B) \in KL_S$ and A s.t. $A \in CPK$ or $A \notin PK$ then reset $k \leftarrow R_S^{\text{sid}}(B, A, X)$
 add $\langle (\text{sid}, C, S, X, Y, \sigma), k \rangle$ to T_H and output k

Fig. 5. 3DH: Environment's view of ideal-world interaction (Game 7)

Theorem 2. *Protocol $HMQR$ shown in Fig. 6 realizes $\mathcal{F}_{\text{khAKE}}$ if the Gap CDH assumption holds and H, H' are random oracles.*

The proof of Theorem 2 follows the template of the proof for the corresponding theorem on 3DH security, i.e. Theorem 1, and we present it in the full version [28].

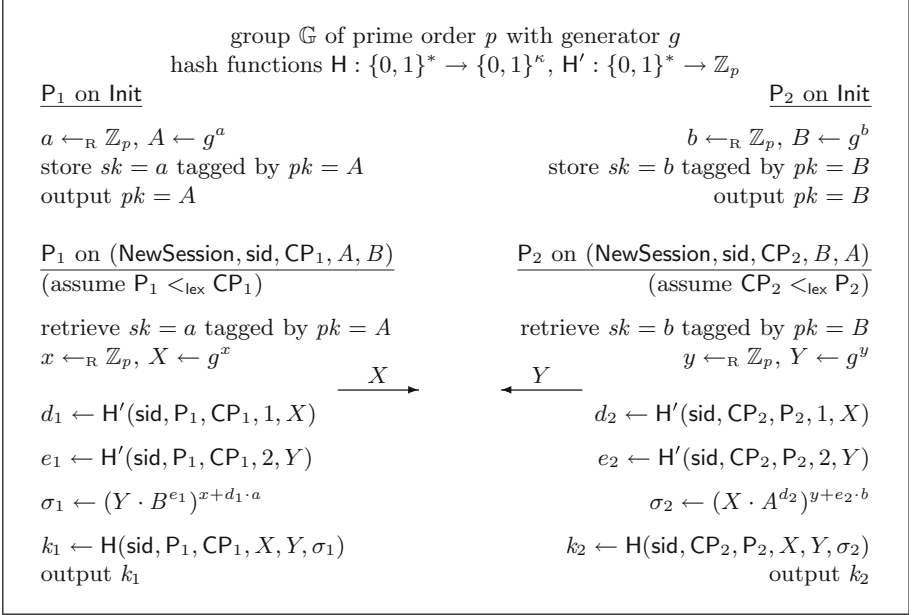


Fig. 6. Protocol HMQV [41]

5 SKEME as Key-Hiding AKE

We present a KEM-based instantiation of the SKEME protocol [39] in Fig. 7. For compliance with the UC notion of AKE modeled by functionality $\mathcal{F}_{\text{khAKE}}$, we derive the session key via a hash involving several additional elements, including a session identifier sid , party identities C and S , public keys A and B , and the transcript X, c, Y, d . We will also use $\{P, CP, A, B, X, c, Y, d, \sigma\}_{\text{ord}}$ to denote $(P, CP, A, B, g^w, c, Z, d, (K, L, Z^w))$ if P plays role = 1, and string $(CP, P, A, B, Z, c, g^w, d, (K, L, Z^w))$ if role = 2. Using this notation each party P can derive its session key as $k \leftarrow H(\text{sid}, \{P, CP, A, B, X, c, Y, d, \sigma\}_{\text{ord}})$.

The security of the protocol relies on two properties of the underlying KEM. First, we assume KEM to be One-Way under Plaintext-Checking-Attack, abbreviated as OW-PCA[31], where the attacker is given access to a Plaintext-Checking Oracle that on input a key K and ciphertext c , it tells if c decapsulates to K under a given KEM key. Second, we require the KEM to be perfectly key-private, namely, given two pairs of private-public keys and a key encapsulation under one of them, one cannot distinguish (information-theoretically) which pair generated that ciphertext. Note the correspondence to the notion of key-hiding PKE [8]. See more details in the full version [28].

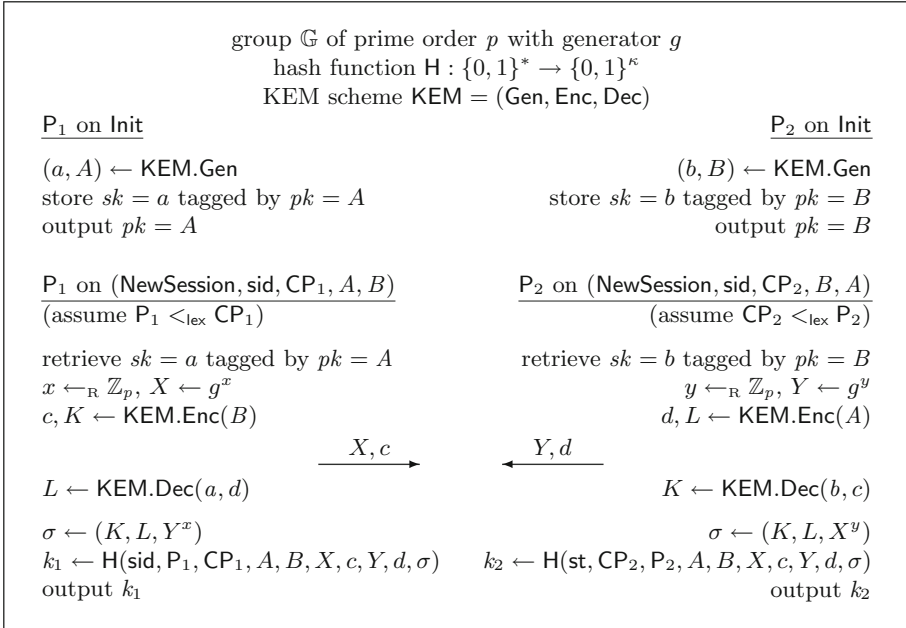


Fig. 7. Protocol SKEME: KEM-authenticated key exchange

Theorem 3. *Protocol SKEME shown in Fig. 7 realizes $\mathcal{F}_{\text{khAKE}}$ if the Gap CDH assumption holds, KEM is a OW-PCA secure and perfect key-private KEM, and H is a random oracle.*

Because of inherent similarities of SKEME and 3DH, the proof of the above theorem follows a similar pattern as the proof of Theorem 1, and we present it in [28].

6 Compiler from Key-Hiding AKE to Asymmetric PAKE

We show that any UC Key-Hiding AKE protocol can be converted to a UC asymmetric PAKE (aPAKE) with a very small computational overhead. We call this AKE-to-aPAKE compiler construction KHAPE, which stands for Key-Hiding Asymmetric PakeE, shown in Fig. 8. The compiler views each party’s AKE inputs, namely its own private key and its counterparty public key, as a single object, an AKE “credential”. The two parties participating in aPAKE, the server and the user, a.k.a. the client, each will have such a credential: The server’s credential contains the server’s private key and the client’s public key, and the client’s credential contains the client’s private key and the server’s public key. Running AKE on such matching pair of inputs would establish a secure shared key, but while the server can store its credential, the client’s only input is her password

and it is not clear how one can derive an AKE credential from a password. Protocol KHAPE enables precisely this derivation: In addition to server’s credential, the server will also store a ciphertext which encrypts, via an ideal cipher, the client’s credential under the user’s password, and the aPAKE protocol consists of server sending that ciphertext to the client, the client decrypting it using the user’s password to obtain its certificate, and using that certificate to run an AKE instance with the server.

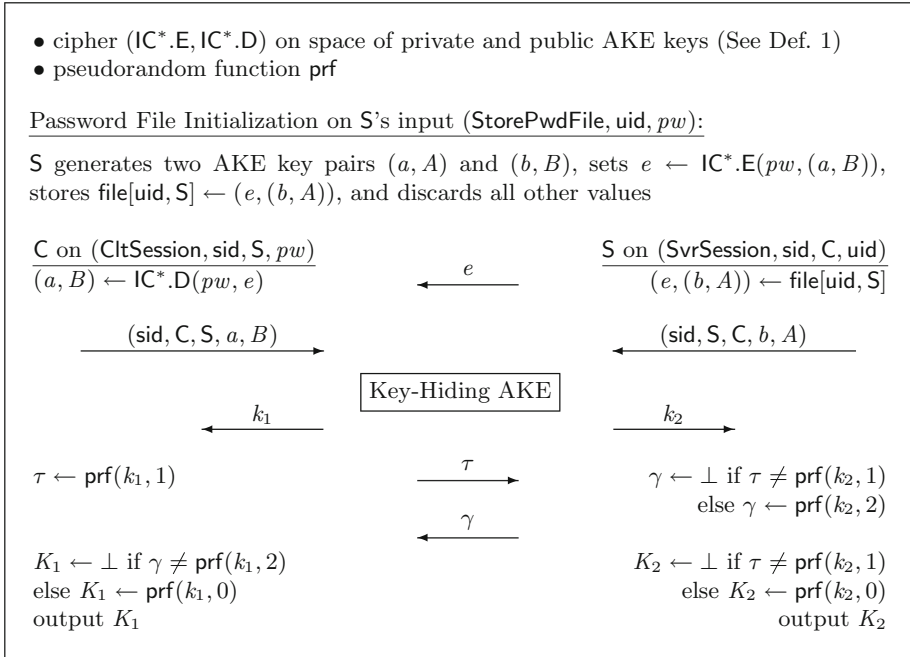


Fig. 8. Protocol KHAPE: Compiler from key-hiding AKE to aPAKE

Reduced-bandwidth variant. In the aPAKE construction in Fig. 8, ciphertext e password-encrypts a pair of the client’s secret key sk_C and the server’s public key pk_S . Without loss of generality every AKE key pair (sk, pk) is generated by the key generation algorithm from uniformly sampled randomness r . The aPAKE construction can be modified so that envelope e password-encrypts only the server’s public key pk_S , while the client derives its private key sk_C using the key generation algorithm on randomness $r \leftarrow H(pw)$ via RO hash H . Note that if key-hiding AKE is either 3DH or HMQV then this amounts to the client setting it’s secret exponent $a \leftarrow H(pw)$ where H maps onto range \mathbb{Z}_q .⁷ This

⁷ If AKE is implemented as SKEME of Sect. 5 then the client must also derive the public key pk_C , since it is used in the key-derivation hash, see Fig. 7.

change does not simplify the construction of the ideal cipher by much because typically the public key is a group element and the private key is a random modular residue, but it reduces the size of ciphertext e . We believe that the security proof for the aPAKE protocol in Fig. 8 can be adjusted to show security of this reduced-bandwidth implementation.

Why we need key-hiding AKE. Note that anyone who observes the credential-encrypting ciphertext e can decrypt it under any password. Each password guess will decrypt e into some credential $cred = (sk_C, pk_S)$, where sk_C is a client's private key and pk_S is a server's public key. Let $cred(pw)$ denote the credential obtained by decrypting e using password pw . For any password guess pw^* the attacker can use credential $cred(pw^*)$ as input to an AKE protocol with the server, but that is equivalent to an on-line password authentication attempt using pw^* as a password guess (see below). Note that the attacker can also either watch or interfere with AKE instances executed by the honest user on credential $cred(pw)$ that corresponds to the correct password pw . Moreover, the attacker w.l.o.g. holds a list of credential candidates $cred(pw_1), \dots, cred(pw_n)$ corresponding to offline password guesses. However, the key-hiding property of AKE implies that even if $cred(pw)$ is on the attacker's list, interfering or watching client's AKE instances cannot help the attacker decide which credential is the one that the client uses. The only way to learn anything from client AKE instances on input $cred(pw)$ would be to engage them using a matching credential, i.e. (sk_S, pk_C) . This is possible if the adversary compromises the server who holds exactly these keys, but otherwise doing so is equivalent to breaking AKE security.

Why we need mutual key confirmation. To handle the server-side attack we needed the key-hiding property of AKE to imply that the only way to decide which keys (sk_S, pk_C) the server uses is to engage in an AKE instance using the matching counterparty keys (sk_C, pk_S) . The key-hiding property provided by 3DH and HMQV, as modeled by functionality $\mathcal{F}_{\text{khAKE}}$, actually does *not* suffice for this by itself. Let the attacker hold a list of n possible decrypted client credentials $cred_i = cred(pw_i) = (a_i, B_i)$ for $i = 1, \dots, n$, and let S hold credential $cred_S = (b, A)$ which matches $cred_i$, i.e. $A = g^{a_i}$ and $B_i = g^b$, which is the case if password guess pw_i matches the correct password pw . If an active attacker chooses x and sends $X = g^x$ to S then it can locally complete the 3DH or HMQV equation using *any* key pair (a_i, B_i) it holds, thus computing n candidate session keys k_i . By 3DH or HMQV correctness, since the i -th client credential matches the server's credential, key k_i equals to the session key k computed by S . Therefore, if S used key k straight away then the attacker could observe that $k_i = k$ and hence that $pw_i = pw$.

However, the fix is simple: To make the server's session key output safe to use, the client must first send a key confirmation message to the server, implemented in Fig. 8 by client's final message τ . This stops the attack because the attacker sending τ uniquely determines one of the keys k_i on its candidate list, and since this succeeds only if $k_i = k$, this attack reduces to an on-line test of a single password guess pw_i , which is unavoidable in a (a)PAKE protocol. A natural

question is if there is no equivalent attack on the client-side, which would be abetted by the client sending a key confirmation message τ . This is not the case because of the following asymmetry: Off-line password guesses give the attacker a list of possible *client-side* credentials, which by AKE rules can be tested against server sessions. However, by the key-hiding property of AKE such credentials are useless in deciding which of them, if any, is used by the honest user. Moreover, since the ciphertext e encrypts only the client-side keys, by the KCI property of the AKE the knowledge of client-side keys is not helpful in breaking the security of AKE instances executed by the honest client on such keys.

Server-to-client key confirmation is needed too, in this case to ensure forward secrecy. Without it, an attacker could choose $Y = g^y$ (in the HMQV or 3DH instantiations) and later, after the session is complete, compromise the server to learn the private key b with which it can compute the session key. The client-to-server key confirmation addresses this issue on the client side.

In addition to ensuring security, key confirmation serves as (explicit) *entity authentication* in this aPAKE construction.

Why we need credential encryption to be an ideal cipher. Note that the attacker can attack the client too, by sending an arbitrary ciphertext to the client, but the ideal cipher property is that the ciphertext commits the attacker to only one choice of key for which the attacker can decide a plaintext: for all other keys the decrypted plaintext will be random.

For the above to work the encryption used to password-encrypt the client credential needs to be an ideal cipher over the space of (private,public) key pairs used in AKE. In all key-hiding AKE protocols examples we discuss in this paper, i.e. 3DH, HMQV, as well as SKEME instantiated with Diffie-Hellman KEM, this message space is $\mathbb{Z}_p \times \mathbb{G}$ where \mathbb{G} is a group of order p . We refer to Sect. 8 for several methods of instantiate an ideal cipher on this space. Here we will assume the implementation of the following form, which is realized by the Elligator2 or Elligator-squared encodings (see Sect. 8).

Definition 1. [(IC*.E, IC*.D) instantiation.] *Let X be the Cartesian product of the space of private keys and the space of public keys in AKE, let IC.E, IC.D be an ideal cipher on n -bit strings, and let map be a (randomized) invertible quasi-bijective map from X to $X' = \{0, 1\}^n$. A randomized 1-1 function $\text{map} : X \rightarrow X'$ is quasi-bijective if there is a negligible statistical difference between a uniform distribution over X' and $x' \leftarrow_R \text{map}(x)$ for random x in X . Instead of a direct ideal cipher on message space X protocol KHAPE in Fig. 8 uses a randomized cipher (IC*.E, IC*.D) on X' where IC*.E(x) outputs IC.E(x') where $x' \leftarrow \text{map}(x; r)$ for random r used by map , and IC*.D(y) outputs $x = \text{map}^{-1}(x')$ where $x' = \text{IC.D}(y)$.*

Comparison with Encrypted Key Exchange of Bellare-Meritt. It is instructive to compare the KHAPE design to that of the “Encrypted Key Exchange” (EKE) construction of Bellare-Meritt [10]. The EKE compiler starts from unauthenticated KE, uses an Ideal Cipher to encrypt each KE protocol message under the password, and this results in UC PAKE in the IC model (see e.g.

[46]). By contrast, our compiler starts from *Authenticated* KE, and uses IC to password-encrypt only the client’s inputs to the AKE protocol, while the protocol messages themselves are exchanged without any change. Just like EKE, our compiler adds only symmetric-key overhead to the underlying KE, but it results in an aPAKE instead of just PAKE. However, just like EKE, it imposes additional requirements on the underlying key exchange protocol: Whereas EKE needs the key exchange to have a “random transcript” property, i.e. KE protocol messages must be random in some message space, in the case of KHAPE the underlying AKE needs to have the key-hiding property we define in Sect. 2. Either condition also relies on an Ideal Cipher (IC) modeling for a non-standard plaintext space: For EKE the IC plaintext space is the space of KE protocol *messages*, while for KHAPE the IC plaintext space is the Cartesian product of the space of private keys and the space of public keys which form AKE protocol *inputs*.

UC aPAKE security model. The UC functionality $\mathcal{F}_{\text{aPAKE}}$ with which we model aPAKE security corresponds to the functionality from Gentry et al. [27] with some slight modifications. The main notational change is that we use a *user account identifier* uid , instead of generic *session identifier* sid , to index password files held by a given server. Functionality $\mathcal{F}_{\text{aPAKE}}$ also includes uni-directional (client-to-server) entity authentication as part of the security definition. $\mathcal{F}_{\text{aPAKE}}$ is described in the full version of the paper [28] where we also discuss several subtle issues involved in UC modeling of tight bounds on adversary’s local computation during an offline dictionary attack.

Theorem 4. *Protocol KHAPE realizes the UC aPAKE functionality $\mathcal{F}_{\text{aPAKE}}$ if the AKE protocol realizes the Key-Hiding AKE functionality $\mathcal{F}_{\text{khAKE}}$, assuming that prf is a secure PRF and (Enc, Dec) is an ideal cipher over message space of private, public key pairs in AKE.*

The proof of the theorem is presented in the full version of the paper [28].

7 Concrete aPAKE Instantiation: KHAPE-HMQV

We include a concrete aPAKE protocol we call KHAPE-HMQV, which results from instantiating protocol KHAPE shown in Sect. 6 with HMQV as the key-hiding AKE (as proved in Sect. 4). The resulting protocol is shown in Fig. 9. It uses only 1 fixed-base exponentiation plus 1 variable-base (multi)exponentiation for each party, and 1 ideal cipher decryption for the client. It has 3 flows if the server initiates and 4 if the client initiates. The communication costs include one group element and a κ -bit key authenticator for both sides plus an ideal cipher encryption of a field element a and another group element B from server to client. Implementations of an ideal cipher over field elements may expand the ciphertext by $\Omega(\kappa)$ bits and require a hash-to-curve operation, see Sect. 8.

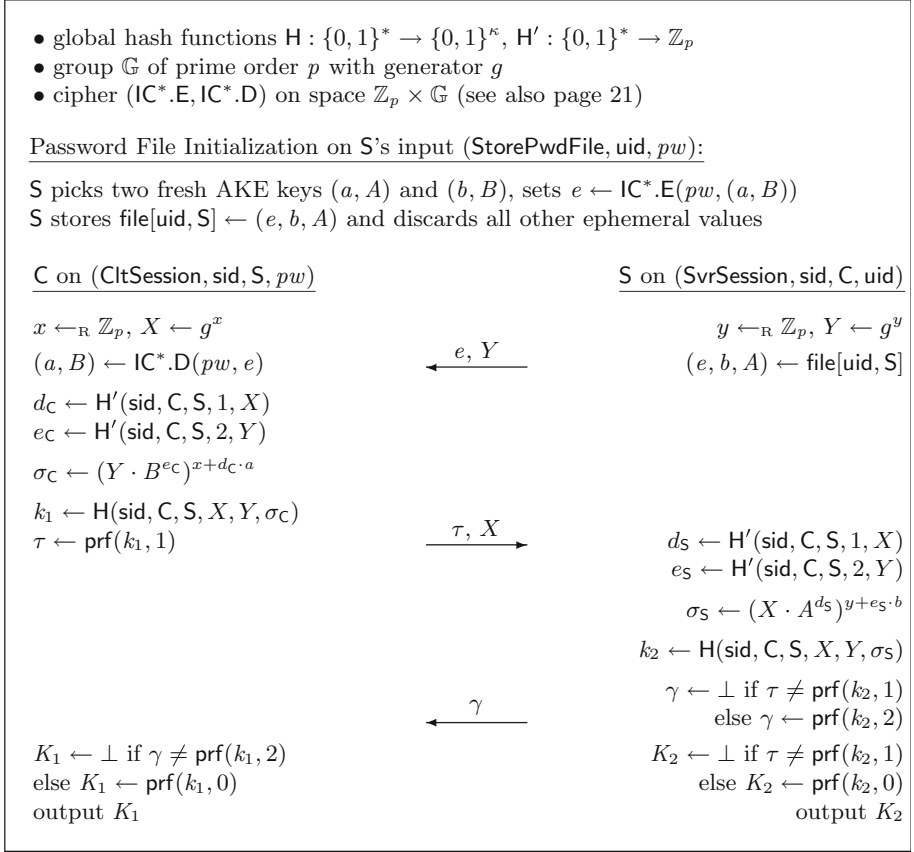


Fig. 9. KHAPE with HMQV: Concrete aPAKE protocol KHAPE-HMQV

While we are showing the protocol with the encryption of credentials done on the server side during password registration (initialization), this can be done interactively by the server sending its public key and the user encrypting it together with its private key under the password (or it can all be done on the client side if the client chooses the server's public key). It is important to highlight that the server needs a random independent pair of private-public keys per user. One optimization is to omit the encryption of the user's private key, and instead derive this key from the password. Our analysis can be adapted to this case.

We note that KHAPE can be made into a Strong aPAKE (saPAKE), secure against pre-computation attacks, using the technique of [37]. Namely, running an OPRF protocol on pw between client and server and deriving the credential encryption key from the output of the OPRF. In addition to providing saPAKE security, the OPRF strengthens the protocol against online client-side attacks (the attacker cannot have a pre-computed list of passwords to try) and it allows for distributing the server through a threshold OPRF. As discussed in the

introduction, the break of the OPRF in the context of KHAPE voids the above benefits but does not endanger the password (a major advantage of KHAPE over OPAQUE).

8 Curve Encodings and Ideal Cipher

8.1 Quasi Bijections

Protocol KHAPE encrypts group elements (server's public key pk_S) using an encryption function modeled as an ideal cipher which works over a space $\{0, 1\}^n$ for some n . Thus, prior to encryption, group elements need to be encoded as bitstrings of length n to which the ideal cipher will be applied. We require such encoding, denoted \mathbf{map} , from G to $\{0, 1\}^n$ to be a bijection (or close to it) so that if e is an encryption of $g \in G$ under password pw , its decryption under a different pw' returns a random element in G . The following definition considers randomized encodings.

Definition 2. A randomized ε -quasi bijection \mathbf{map} with domain A , randomness space $R = \{0, 1\}^\rho$ and range B consists of two algorithms \mathbf{map} and \mathbf{map}^{-1} , $\mathbf{map} : A \times R \rightarrow B$ and $\mathbf{map}^{-1} : B \rightarrow A$ with the following properties:

1. \mathbf{map}^{-1} is deterministic and for all $a \in A, r \in R, \mathbf{map}^{-1}(\mathbf{map}(a, r)) = a$;
2. \mathbf{map} maps the uniform distribution on $A \times R$ to a distribution on B that is ε -close to uniform.

The term ε -close refers to a statistical distance of at most ε between the two distributions. It can also be used in the sense of computational indistinguishability, e.g., if implementing randomness using a PRG. To accommodate bijections whose randomized \mathbf{map} from A to B may exceed a given time bound in some inputs, one can consider the range of \mathbf{map} to include an additional element \perp to which such inputs are mapped. A simpler way is to define that such inputs are mapped to a fixed element in B . The probability of inputs mapped to that value is already accounted for in the statistical distance bound ε . We use *quasi bijection* without specifying ε when we assume this value to be negligible.

Quasi bijections from field elements to bitstrings. We are interested in quasi-bijective encoding into the set $\{0, 1\}^n$ over which the IC encryption works. Most mappings presented below have a field \mathbb{Z}_q as the range, in which case a further transformation (preserving quasi-bijectiveness) may be needed. Note that when representing elements of \mathbb{Z}_q as n -bit numbers for $n = \lceil \log q \rceil$, the uniform distribution on \mathbb{Z}_q is ε -close to the uniform distribution over $\{0, 1\}^n$ for $\varepsilon = (2^n \bmod q)/q$. So when q is very close to 2^n , one can use the bit representation of field elements directly, and this is the case for many of the standardized elliptic curves. When this is not the case, one maps $u \in \mathbb{Z}_q$ to a $(n+k)$ -bit integer selected as $u + tq$ for t randomly chosen as a non-negative integer $< (2^{n+k} - u)/q$. The resulting distribution is 2^{-k} -close to the uniform distribution over $\{0, 1\}^{n+k}$.

8.2 Implementing Quasi-Bijective Encodings

We focus on the case where G is an elliptic curve. There is a large variety of well-studied quasi-bijective encodings in the literature (cf. [11, 16, 26, 50, 53]). We survey some representative examples for elliptic curve groups $EC(q)$ over fields of large prime-order q .

Note that we use both directions of these encodings in KHAPE: From pk_S to a bitstring when encrypting pk_S at the time of password registration, and from a bitstring to a curve point when the client decrypts pk_S . This means that the performance of the latter operation is more significant for the efficiency of the protocol. Fortunately this is always the more efficient direction, even though the other direction is quite efficient too for the maps discussed below.

Elligator-squared [38, 53]. This method applies to most elliptic curves and accommodates ε -quasi bijections for the *whole set of curve points* with negligible values of ε .

Curve points are encoded as a pair of field elements $(u, v) \in \mathbb{Z}_q^2$. There is a deterministic function f from \mathbb{Z}_q to EC such that $P \in EC$ is represented by (u, v) if and only if $P = f(u) + f(v)$. Given a point P there is a randomized procedure R_f that returns such encoding (u, v) .

In [53] (Theorem 1), it is proven that for suitable choices of f , R_f is an ε -quasi bijection into $(\mathbb{Z}_q)^2$, with $\varepsilon = O(q^{-1/2})$ (see Definition 2). Since u, v are field elements, a further bijection into bitstrings may be needed as specified in Sect. 8.1.

In [38], the above construction is improved by allowing both u and v to be represented directly as bit strings: u as a string of $\lfloor q \rfloor$ bits and v can be shortened even further (the amount of shortening increases the statistical distance for the quasi bijection from EC to the distribution of bitstrings (u, v)). This encoding uses two functions f, g where a point P is recovered from (u, v) as $P = f(u) + g(v)$ (in this case, function g can be simply $g(v) = v \cdot P$).

The performance of Elligator-squared depends on the functions f, g whose cost with typical instantiations (e.g., Elligator, SWU) is dominated by a single base-field exponentiation at the cost of a fraction ($\approx 10\text{--}15\%$) of a scalar multiplication. Implementing $g(v) = v \cdot P$ is also a low-cost option (also allowing to shorten v [38]). The cost of the inverse map, from a curve point to its bitstring encoding, for the curves analyzed in [53] is 3 base-field exponentiations.

Elligator2. This mapping from [11] is of more restricted applicability than Elligator-squared as it applies to a smaller set of curves (e.g., it requires an element of order 2). Yet, this class includes some of the common curves used in practice, particularly Curve25519. Elligator2 defines an injective mapping between the integers $\{0, \dots, (q-1)/2\}$ and (about) half of the elements in the curve. To be used in our setting, it means that when generating a pair $(sk_S, pk_{S=g^{sk_S}})$ for the server during password registration, the key generation procedure will choose a random sk_S and will test if the resultant pk_S has a valid encoding under Elligator2. If so, it will keep this pair, otherwise it will choose another random pair and repeat until a representable point is found. The expected number of trials is

2 and the testing procedure is very efficient (and only used during registration, not for login).

The advantages of Elligator2 include the use of a single field element as a point representation (which requires further expansion into a bit string only if q is not close to 2^n) and the map is injective, hence quasi-bijective with $\varepsilon = 0$ over the subset of encodable curve elements. Both directions of the map are very efficient, costing about a single base-field exponentiation (a fraction of the cost of a scalar multiplication).

Detailed implementation information for the components of the above transforms is found in [11, 25, 54]. See [7] for some comparison between Elligator2 and Elligator-squared.

8.3 Ideal Cipher Constructions

Protocol KHAPE uses an ideal cipher to encrypt group elements, specifically a pair (sk_C, pk_S) where both elements are encoded as bitstrings to fit the ideal cipher interface as described in previous subsections. Thus, we consider the input to the encryption simply as a bitstring of a given fixed length, and require implementations of ideal ciphers of sufficiently long block length. For example, the combined input length for curves of 256 bits ranges between 512 and 1024 bits. Constructions of encryption schemes that are indistinguishable from an ideal cipher have been investigated extensively in the literature. Techniques include domain extension mechanisms (e.g., to expand the block size for block ciphers, including AES) [19], Feistel networks and constructions from random oracles [20, 23, 32], dedicated constructions such as those based on iterated Even-Mansour and key alternating ciphers [6, 22, 24, 24], and basic components such as wide-input (public) random permutations [12, 13, 21]. A recent technique by McQuoid et al. [46], builds a dedicated transform that can replace the ideal cipher in cases where encryption is “one-time”, namely, keys (or cipher instances) are used to encrypt a single message (as in our protocols). They build a very efficient transform using a random oracle with just two Feistel rounds. A dedicated analysis for the use of this technique in our context is left for future work.

Acknowledgment. We thank the anonymous referees for their insightful comments.

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>
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>
3. Abdalla, M., Barbosa, M., Bradley, T., Jarecki, S., Katz, J., Xu, J.: Universally composable relaxed password authenticated key exchange. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 278–307. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_10

4. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Efficient two-party password-based key exchange protocols in the UC framework. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 335–351. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79263-5_22
5. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_14
6. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indistinguishability of key-alternating ciphers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 531–550. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_29
7. Aranha, D.F., Fouque, P.-A., Qian, C., Tibouchi, M., Zapalowicz, J.-C.: Binary elligator squared. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 20–37. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13051-4_2
8. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
9. 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
10. Bellare, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: IEEE Computer Society Symposium on Research in Security and Privacy - S&P 1992, pp. 72–84. IEEE (1992)
11. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: ACM Conference on Computer and Communications Security - CCS 2013 (2013)
12. Bernstein, D.J., et al.: Gimli: a cross-platform permutation. Cryptology ePrint Archive, Report 2017/630 (2017). <http://eprint.iacr.org/2017/630>
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_19
14. 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
15. Bradley, T., Jarecki, S., Xu, J.: Strong asymmetric PAKE based on trapdoor CKEM. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 798–825. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_26
16. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indistinguishable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_13
17. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: IEEE Symposium on Foundations of Computer Science - FOCS 2001, pp. 136–145. IEEE (2001)
18. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_28

19. 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
20. Dachman-Soled, D., Katz, J., Thiruvengadam, A.: 10-round Feistel is indifferentiable from an ideal cipher. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 649–678. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_23
21. Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: The design of Xoodoo and Xooff, pp. 1–38 (2018)
22. Dai, Y., Seurin, Y., Steinberger, J., Thiruvengadam, A.: Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: five rounds are necessary and sufficient. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 524–555. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_18
23. Dai, Y., Steinberger, J.: Indifferentiability of 8-round Feistel networks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 95–120. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_4
24. Dodis, Y., Stam, M., Steinberger, J., Liu, T.: Indifferentiability of confusion-diffusion networks. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 679–704. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_24
25. Faz-Hernandez, A., Scott, S., Sullivan, N., Wahby, R., Wood, C.: Hashing to elliptic curves draft-IRTF-CFRG-hash-to-curve, June 2020. <https://datatracker.ietf.org/doc/draft-irtf-cfrg-hash-to-curve/>
26. Fouque, P.-A., Joux, A., Tibouchi, M.: Injective encodings to elliptic curves. In: Boyd, C., Simpson, L. (eds.) ACISP 2013. LNCS, vol. 7959, pp. 203–218. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39059-3_14
27. 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 (2006). https://doi.org/10.1007/11818175_9
28. Gu, Y., Jarecki, S., Krawczyk, H.: KHAFE: Asymmetric PAKE from Key-Hiding Key Exchange. IACR Cryptology ePrint Archive, June 2021. <http://eprint.iacr.org/2021>
29. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. (TISSEC) **2**(3), 230–268 (1999)
30. Hao, F., Shahandashti, S.F.: The SPEKE protocol revisited. Cryptology ePrint Archive, Report 2014/585 (2014). <http://eprint.iacr.org/2014/585>
31. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. Cryptology ePrint Archive, Report 2017/604 (2017). <https://eprint.iacr.org/2017/604>
32. Holenstein, T., Künzler, R., Tessaro, S.: The equivalence of the random oracle model and the ideal cipher model, revisited. In: STOC 2011 (2011)
33. 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 2018. LNCS, vol. 11035, pp. 485–504. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_26
34. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC 1989, pp. 44–61 (1989)

35. Jablon, D.P.: Extended password key exchange protocols immune to dictionary attacks. In: 6th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 1997), pp. 248–255, Cambridge, MA, USA, 18–20 June 1997. IEEE Computer Society (1997)
36. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: TOPPSS: cost-minimal password-protected secret sharing based on threshold OPRF. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 39–58. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_3
37. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 456–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_15
38. Kim, T., Tibouchi, M.: Invalid curve attacks in a GLS setting. In: Tanaka, K., Suga, Y. (eds.) IWSEC 2015. LNCS, vol. 9241, pp. 41–55. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22425-1_3
39. Krawczyk, H.: SKEME: a versatile secure key exchange mechanism for internet. In: 1996 Internet Society Symposium on Network and Distributed System Security (NDSS), pp. 114–127 (1996)
40. Krawczyk, H.: SIGMA: the ‘SIGn-and-MAC’ approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_24
41. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
42. Krawczyk, H., Bourdrez, D., Lewi, K., Wood, C.: The opaque asymmetric pake protocol, draft-IRTF-CFRG-opaque (2021). <https://datatracker.ietf.org/doc/draft-irtf-cfrg-opaque/>
43. MacKenzie, P.: On the security of the SPEKE password-authenticated key exchange protocol. Cryptology ePrint Archive, Report 2001/057 (2001). <http://eprint.iacr.org/2001/057>
44. Marlinspike, M.: Simplifying OTR deniability (2013). <https://signal.org/blog/simplifying-otr-deniability/>
45. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol (2016). <https://signal.org/docs/specifications/x3dh/>
46. McQuoid, I., Rosulek, M., Roy, L.: Minimal symmetric PAKE and 1-out-of-n OT from programmable-once public functions. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) CCS 2020: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, 9–13 November 2020. <https://eprint.iacr.org/2020/1043>
47. NIST Information Technology Lab. Post-quantum cryptography. <https://csrc.nist.gov/projects/post-quantum-cryptography>
48. Pointcheval, D., Wang, G.: VTBPEKE: verifier-based two-basis password exponential key exchange. In ASIACCS 2017, pp. 301–312. ACM Press (2017)
49. Schmidt, J.: Requirements for password-authenticated key agreement (PAKE) schemes, April 2017. <https://tools.ietf.org/html/rfc8125>
50. Shallue, A., van de Woestijne, C.: Construction of rational points on elliptic curves over finite fields. In: ANTS (2006)
51. Shoup, V.: Security analysis of SPAKE2+. IACR Cryptol. ePrint Arch. **2020**, 313 (2020)

52. Sullivan, N., Krawczyk, H., Friel, O., Barnes, R.: OPAQUE with TLS 1.3, draft-sullivan-tls-opaque, February 2021. <https://datatracker.ietf.org/doc/draft-sullivan-tls-opaque/>
53. Tibouchi, M.: Elligator squared: uniform points on elliptic curves of prime order as uniform random strings. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 139–156. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_10
54. Wahby, R.S., Boneh, D.: Fast and simple constant-time hashing to the BLS12-381 elliptic curve, no. 4, pp. 154–179 (2019). <https://tches.iacr.org/index.php/TCHES/article/view/8348>

Author Index

- Agrawal, Shweta 208, 239
Aqeel, Waqar 641
Asharov, Gilad 610
Attema, Thomas 65
- Baum, Carsten 92
Block, Alexander R. 123
Boyle, Elette 487
- Chandrasekaran, Balakrishnan 641
Chen, Rongmao 270
Chen, Yilei 334
Chen, Yu 179
Choudhuri, Arka Rai 394
Couteau, Geoffroy 487
Cramer, Ronald 65
- Dinur, Itai 517
Dodis, Yevgeniy 548
- Fehr, Serge 65
Feng, Hanwen 3
- Gilboa, Niv 487
Goldfeder, Steven 517
Goyal, Rishab 208
Gu, Yanqi 701
Guo, Siyao 548
- Halevi, Tzipora 517
Han, Shuai 670
Holmgren, Justin 123
Huang, Xinyi 270
- Ishai, Yuval 487, 517
- Jager, Tibor 670
Jain, Abhishek 394
Jarecki, Stanislaw 701
Jin, Zhengzhong 394
- Kelkar, Mahimna 517
Kerber, Thomas 364
Kiayias, Aggelos 364
- Kiltz, Eike 670
Kohl, Lisa 487
Kohlweiss, Markulf 364
Komargodski, Ilan 579, 610
Krawczyk, Hugo 701
- Liang, Xiao 34
Lin, Wei-Kai 579, 610
Liu, Shengli 670
Liu, Tianren 454
Lombardi, Alex 334
- Ma, Fermi 334
Maggs, Bruce 641
Maitra, Monosij 239
Malozemoff, Alex J. 92
- Pan, Jiaxin 179, 670
Pandey, Omkant 34
- Quach, Willy 334, 424
- Riepel, Doreen 670
Rosen, Alon 123
Rosen, Marc B. 92
Rothblum, Ron D. 123
- Schäge, Sven 670
Scholl, Peter 92, 487
Sharma, Vivek 517
Shi, Elaine 610, 641
Soni, Pratik 123
Stephens-Davidowitz, Noah 548
- Tang, Qiang 3
Tessaro, Stefano 454
Tomida, Junichi 208
- Vaikuntanathan, Vinod 454
Vempati, Narasimha Sai 239
- Wang, Baosheng 270
Wang, Yi 270
Wang, Yuyu 179

Waters, Brent [424](#)
Wee, Hoeteck [155](#)
Wichs, Daniel [424](#)

Xie, Zhiye [548](#)

Yamada, Shota [239](#)
Yang, Guomin [270](#)
Yung, Moti [270](#)

Zaverucha, Greg [517](#)
Zhandry, Mark [303](#)