

Alexandra Boldyreva  
Vladimir Kolesnikov (Eds.)

LNCS 13941

# Public-Key Cryptography – PKC 2023

26th IACR International Conference  
on Practice and Theory of Public-Key Cryptography  
Atlanta, GA, USA, May 7–10, 2023  
Proceedings, Part II

2  
Part II



 Springer

# Lecture Notes in Computer Science

13941

## Founding Editors

Gerhard Goos  
Juris Hartmanis

## Editorial Board Members

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

Wen Gao, *Peking University, Beijing, China*

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

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

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

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


Alexandra Boldyreva · Vladimir Kolesnikov  
Editors

# Public-Key Cryptography – PKC 2023

26th IACR International Conference  
on Practice and Theory of Public-Key Cryptography  
Atlanta, GA, USA, May 7–10, 2023  
Proceedings, Part II

*Editors*

Alexandra Boldyreva  
Georgia Institute of Technology  
Atlanta, GA, USA

Vladimir Kolesnikov   
Georgia Institute of Technology  
Atlanta, GA, USA

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-031-31370-7

ISBN 978-3-031-31371-4 (eBook)

<https://doi.org/10.1007/978-3-031-31371-4>

© International Association for Cryptologic Research 2023

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 26th International Conference on Practice and Theory of Public-Key Cryptography (PKC 2023) was held in Atlanta, Georgia, USA on May 7–10, 2023. It was sponsored by the International Association for Cryptologic Research (IACR).

The conference received 183 submissions, reviewed by the Program Committee of 49 cryptography experts working with 142 external reviewers. The reviewing process took 2.5 months and resulted in selecting 50 papers to appear in PKC 2023.

Papers were reviewed in the usual double-blind fashion. Program committee members were limited to two submissions, and their submissions were scrutinized more closely. The two program chairs were not allowed to submit papers.

The Program Committee recognized two papers and their authors. “The Hidden Number Problem with Small Unknown Multipliers: Cryptanalyzing MEGA in Six Queries and Other Applications,” by Nadia Heninger and Keegan Ryan, and “Post-Quantum Anonymity of Kyber”, by Varun Maram and Keita Xagawa, were selected Best Papers of the conference.

PKC 2023 welcomed Chris Peikert (University of Michigan) as the invited speaker.

The PKC Test-of-Time Award (ToT) recognizes outstanding and influential papers published in PKC about 15 years prior. The inaugural PKC Test of Time Award was given in PKC 2019 for papers published in the conference’s initial years of the early 2000s and late 1990s. In 2023, the ToT committee, consisting of Alexandra Boldyreva, Goichiro Hanaoka, Vlad Kolesnikov, Moti Yung, and Yuliang Zheng, considered papers published in PKC 2006–2008 for the award. The committee selected the PKC 2008 paper “Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption” by Benoît Libert and Damien Vergnaud for the Test-of-Time award.

PKC is the main IACR-sponsored conference with an explicit focus on public-key cryptography. It is a remarkable undertaking, only possible due to the hard work and significant contributions of many people. We would like to express our sincere gratitude to the authors of all submitted works, as well as to the PC and external reviewers, session chairs and presenters. Additionally, we would like to thank the following people and organizations for helping make PKC 2023 a success:

- Joseph Jaeger and Daniel Genkin – PKC 2023 General Chairs,
- Chris Peikert – invited speaker,
- Kay McKelly and Kevin McCurley – all things technical behind the scenes,
- Ellen Kolesnikova – design of the PKC 2023 logo,
- the team at Springer,
- Georgia Tech Hotel and Conference Center,
- Georgia Aquarium,
- School of Cybersecurity and Privacy at Georgia Tech - the academic home of the PKC 2023 Program and General Chairs.

We would also like to thank our sponsors: Google (platinum), Starkware (silver), Amazon AWS (silver), and Algorand (bronze). 2022 and 2023 were difficult years in the

tech industry, making sponsors' contributions ever more valued. Their generous support covered several student travel stipends and helped minimize registration fees, including half-priced registration for all students.

Lastly, a big thanks to everyone who attended PKC 2023 in Atlanta. We hope you enjoyed the conference and the warm welcome of our city and university.



May 2023

Alexandra Boldyreva  
Vlad Kolesnikov

# Organization

## General Chairs

Daniel Genkin	Georgia Tech, USA
Joseph Jaeger	Georgia Tech, USA

## Program Committee Chairs

Alexandra Boldyreva	Georgia Tech, USA
Vladimir Kolesnikov	Georgia Tech, USA

## Steering Committee

Masayuki Abe	NTT, Japan
Jung Hee Cheon	Seoul National University, Korea
Yvo Desmedt	University of Texas at Dallas, USA
Goichiro Hanaoka	AIST, Japan
Aggelos Kiayias	University of Edinburgh, UK
Tanja Lange	Eindhoven University of Technology, Netherlands
David Pointcheval	École Normale Supérieure, France
Moti Yung (Secretary)	Google Inc. & Columbia University, USA
Yuliang Zheng (Chair)	University of Alabama at Birmingham, USA

## Program Committee

Ghada Almashaqbeh	University of Connecticut, USA
Nuttapong Attrapadung	AIST, Japan
Carlo Blundo	Università degli Studi di Salerno, Italy
Katharina Boudgoust	Aarhus University, Denmark
Dario Catalano	Università di Catania, Italy
Suvradip Chakraborty	ETH Zurich, Switzerland
Shan Chen	Southern University of Science & Technology, China
Jean Paul Degabriele	Technology Innovation Institute, UAE
Chaya Ganesh	Indian Institute of Science, India



Sean Hallgren	Penn State University, USA
David Heath	University of Illinois Urbana-Champaign, USA
Kristina Hostakova	ETH Zürich, Switzerland
Sorina Ionica	Université de Picardie Jules Verne, France
Stanislaw Jarecki	University of California, Irvine, USA
Shuichi Katsumata	AIST and PQShield Ltd., Japan
Kaoru Kurosawa	AIST, Japan
Tancrède Lepoint	Amazon, USA
Christian Majenz	Technical University of Denmark, Denmark
Daniel Masny	Meta, USA
Ryo Nishimaki	NTT Social Informatics Laboratories, Japan
Adam O'Neill	UMass Amherst, USA
Charalampos Papamanthou	Yale University, USA
Alain Passelègue	Inria and ENS Lyon, France
Sikhar Patranabis	IBM Research India, India
Alice Pellet-Mary	CNRS and Université de Bordeaux, France
Edoardo Persichetti	Florida Atlantic University, USA
Rachel Player	Royal Holloway, University of London, UK
David Pointcheval	ENS, Paris, France
Antigoni Polychroniadou	JPMorgan AI Research, USA
Willy Quach	Northeastern University, USA
Elizabeth Quaglia	Royal Holloway, University of London, UK
Adeline Roux-Langlois	Normandie Univ, GREYC, France
John Schanck	Mozilla, USA
Peter Scholl	Aarhus University, Denmark
Dominique Schröder	FAU Erlangen-Nürnberg, Germany
Peter Schwabe	MPI-SP & Radboud University, Netherlands
Jae Hong Seo	Hanyang University, Korea
Abhi Shelat	Northeastern University, USA
Akira Takahashi	University of Edinburgh, UK
Keisuke Tanaka	Tokyo Institute of Technology, Japan
Jean-Pierre Tillich	Inria, France
Frederik Vercauteren	KU Leuven, Belgium
Damien Vergnaud	Sorbonne Université, France
Ivan Visconti	University of Salerno, Italy
Benjamin Wesolowski	CNRS and University of Bordeaux, France
David Wu	UT Austin, USA
Kevin Yeo	Google and Columbia University, USA
Mark Zhandry	NTT Research & Princeton University, USA
Vassilis Zikas	Purdue University, USA

## Additional Reviewers

Behzad Abdolmaleki  
Calvin Abou Haidar  
Ojaswi Acharya  
Gorjan Alagic  
Gennaro Avitabile  
Arnab Bag  
Shi Bai  
Magali Bardet  
Hugo Beguinet  
Fabrice Benhamouda  
Loris Bergerat  
Ward Beullens  
Olivier Blazy  
Maxime Bombar  
Cecilia Boschini  
Vincenzo Botta  
Samuel Bouaziz-Ermann  
Charles Bouillaguet  
Nicholas Brandt  
Lennart Braun  
Matteo Campanelli  
André Chailloux  
Rohit Chatterjee  
Jesus-Javier Chi-Dominguez  
Hien Chu  
Heewon Chung  
Michele Ciampi  
Jean-Sébastien Coron  
Anamaria Costache  
Baptiste Cottier  
Jan-Pieter D'Anvers  
Pratish Datta  
Gareth T. Davies  
Paola De Perthuis  
Jean-Christophe Deneuille  
Julien Devevey  
Mario Di Raimondo  
Javad Doliskani  
Keita Emura  
Andreas Erwig  
Daniel Escudero  
Andre Esser  
Pouria Fallahpour  
Antonio Faonio  
Joël Felderhoff  
Weiqi Feng  
Rune Fiedler  
Georgios Fotiadis  
Tako Boris Fouotsa  
Georg Fuchsbauer  
Clemente Galdi  
Romain Gay  
Robin Geelen  
Paul Gerhart  
Lenaïck Gouriou  
Mohammad Hajiabadi  
Erin Hales  
Mickaël Hamdad  
Patrick Harasser  
Keitaro Hashimoto  
Sorina Ionica  
Vincenzo Iovino  
Aayush Jain  
Christian Janson  
Corentin Jeudy  
Saqib Kakvi  
Daniel Kales  
Harish Karthikeyan  
Julia Kastner  
Mojtaba Khalili  
Hamidreza Khoshakhlagh  
Ryo Kikuchi  
Dongwoo Kim  
Elena Kirshanova  
Fuyuki Kitagawa  
David Kohel  
Sebastian Kolby  
Walter Krawec  
Mikhail Kudinov  
Péter Kutas  
Roman Langrehr  
Mario Larangeira  
Changmin Lee  
Antonin Leroux  
Andrea Lesavourey  
Varun Madathil

Lorenzo Magliocco  
Jules Maire  
Monosij Maitra  
Takahiro Matsuda  
Liam Medley  
Kelsey Melissaris  
Hart Montgomery  
Ngoc Khanh Nguyen  
Ky Nguyen  
Thi Thu Quyen Nguyen  
Phong Nguyen  
Ruben Niederhagen  
Koji Nuida  
Tapas Pal  
Kunjai Panchal  
Mahak Pancholi  
Lorenz Panny  
Robi Pedersen  
Lucas Prabel  
Thomas Prest  
Sihang Pu  
Krijn Reijnders  
Mahshid Riahinia  
Doreen Riepel  
Felix Rohrbach  
Mélissa Rossi  
Olga Sanina  
Paolo Santini

André Schrottenloher  
Robert Schädlich  
Yixin Shen  
Mark Simkin  
Animesh Singh  
Sayani Sinha  
Luisa Siniscalchi  
Christoph Striecks  
Atsushi Takayasu  
Debadrita Talapatra  
Aravind Thyagarajan  
Junichi Tomida  
Toi Tomita  
Monika Trimoska  
Damien Vidal  
Chenkai Wang  
Yohei Watanabe  
Christian Weinert  
Weiqiang Wen  
Keita Xagawa  
Shota Yamada  
Takashi Yamakawa  
Yibin Yang  
Kazuki Yoneyama  
Yusuke Yoshida  
Bor de Kock  
Rafael del Pino  
Wessel van Woerden

## Contents – Part II

### Homomorphic Cryptography and Other Topics

On Homomorphic Secret Sharing from Polynomial-Modulus LWE .....	3
<i>Thomas Attema, Pedro Capitão, and Lisa Kohl</i>	
Discretization Error Reduction for High Precision Torus Fully Homomorphic Encryption .....	33
<i>Kang Hoon Lee and Ji Won Yoon</i>	
Verifiable Capacity-Bound Functions: A New Primitive from Kolmogorov Complexity: (Revisiting Space-Based Security in the Adaptive Setting) .....	63
<i>Giuseppe Ateniese, Long Chen, Danilo Francati, Dimitrios Papadopoulos, and Qiang Tang</i>	
A Holistic Approach Towards Side-Channel Secure Fixed-Weight Polynomial Sampling .....	94
<i>Markus Krausz, Georg Land, Jan Richter-Brockmann, and Tim Güneysu</i>	

### MPC

Private Polynomial Commitments and Applications to MPC .....	127
<i>Rishabh Bhadauria, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Wenxuan Wu, and Yupeng Zhang</i>	
Credibility in Private Set Membership .....	159
<i>Sanjam Garg, Mohammad Hajiabadi, Abhishek Jain, Zhengzhong Jin, Omkant Pandey, and Sina Shiehian</i>	
Improved Private Set Intersection for Sets with Small Entries .....	190
<i>Dung Bui and Geoffroy Couteau</i>	
Pseudorandom Correlation Functions from Variable-Density LPN, Revisited .....	221
<i>Geoffroy Couteau and Clément Ducros</i>	
Threshold Private Set Intersection with Better Communication Complexity ....	251
<i>Satrajit Ghosh and Mark Simkin</i>	

**Encryption**

Almost Tightly-Secure Re-randomizable and Replayable CCA-Secure  
Public Key Encryption ..... 275  
*Antonio Faonio, Dennis Hofheinz, and Luigi Russo*

Multi-authority ABE for Non-monotonic Access Structures ..... 306  
*Miguel Ambrona and Romain Gay*

Multi-instance Secure Public-Key Encryption ..... 336  
*Carlo Brunetta, Hans Heum, and Martijn Stam*

Unidirectional Updatable Encryption and Proxy Re-encryption from DDH ..... 368  
*Peihan Miao, Sikhar Patranabis, and Gaven Watson*

Backward-Leak Uni-Directional Updatable Encryption  
from (Homomorphic) Public Key Encryption ..... 399  
*Yao Jiang Galteland and Jiaxin Pan*

Functional Encryption Against Probabilistic Queries: Definition,  
Construction and Applications ..... 429  
*Geng Wang, Shi-Feng Sun, Zhedong Wang, and Dawu Gu*

**ZK I**

A Generic Transform from Multi-round Interactive Proof to NIZK ..... 461  
*Pierre-Alain Fouque, Adela Georgescu, Chen Qian,  
Adeline Roux-Langlois, and Weiqiang Wen*

Fine-Grained Verifier NIZK and Its Applications ..... 482  
*Xiangyu Liu, Shengli Liu, Shuai Han, and Dawu Gu*

Zero-Knowledge Arguments for Subverted RSA Groups ..... 512  
*Dimitris Kolonelos, Mary Maller, and Mikhail Volkhov*

Dew: A Transparent Constant-Sized Polynomial Commitment Scheme ..... 542  
*Arasu Arun, Chaya Ganesh, Satya Lokam, Tushar Mopuri,  
and Sriram Sridhar*

**IO and ZK II**

Non-Interactive Publicly-Verifiable Delegation of Committed Programs ..... 575  
*Riddhi Ghosal, Amit Sahai, and Brent Waters*

Laconic Function Evaluation for Turing Machines ..... 606  
*Nico Döttling, Phillip Gajland, and Giulio Malavolta*

A Map of Witness Maps: New Definitions and Connections ..... 635  
*Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs*

Structure-Preserving Compilers from New Notions of Obfuscations ..... 663  
*Matteo Campanelli, Danilo Francati, and Claudio Orlandi*

**Author Index** ..... 695

# Contents – Part I

## Post-quantum Cryptography

Post-quantum Anonymity of Kyber .....	3
<i>Varun Maram and Keita Xagawa</i>	
QCCA-Secure Generic Transformations in the Quantum Random Oracle Model .....	36
<i>Tianshu Shan, Jiangxia Ge, and Rui Xue</i>	
A Thorough Treatment of Highly-Efficient NTRU Instantiations .....	65
<i>Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, Gregor Seiler, and Dominique Unruh</i>	
A Lightweight Identification Protocol Based on Lattices .....	95
<i>Samed Düzlülü, Juliane Krämer, Thomas Pöppelmann, and Patrick Struck</i>	
POLKA: Towards Leakage-Resistant Post-quantum CCA-Secure Public Key Encryption .....	114
<i>Clément Hoffmann, Benoît Libert, Charles Momin, Thomas Peters, and François-Xavier Standaert</i>	

## Attacks

The Hidden Number Problem with Small Unknown Multipliers: Cryptanalyzing MEGA in Six Queries and Other Applications .....	147
<i>Nadia Heninger and Keegan Ryan</i>	
Hull Attacks on the Lattice Isomorphism Problem .....	177
<i>Léo Ducas and Shane Gibbons</i>	
A Key-Recovery Attack Against Mitaka in the $t$ -Probing Model .....	205
<i>Thomas Prest</i>	

## Signatures

Hardening Signature Schemes via Derive-then-Derandomize: Stronger Security Proofs for EdDSA .....	223
<i>Mihir Bellare, Hannah Davis, and Zijing Di</i>	

Security Analysis of RSA-BSSA .....	251
<i>Anna Lysyanskaya</i>	
Extendable Threshold Ring Signatures with Enhanced Anonymity .....	281
<i>Gennaro Avitabile, Vincenzo Botta, and Dario Fiore</i>	
Tracing a Linear Subspace: Application to Linearly-Homomorphic Group Signatures .....	312
<i>Chloé Héban, David Pointcheval, and Robert Schädlich</i>	
<b>Isogenies</b>	
SCALLOP: Scaling the CSI-FiSh .....	345
<i>Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski</i>	
Round-Optimal Oblivious Transfer and MPC from Computational CSIDH .....	376
<i>Saikrishna Badrinarayanan, Daniel Masny, Pratyay Mukherjee, Sikhar Patranabis, Srinivasan Raghuraman, and Pratik Sarkar</i>	
Generic Models for Group Actions .....	406
<i>Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel</i>	
<b>Crypto for Crypto</b>	
CRAFT: Composable Randomness Beacons and Output-Independent Abort MPC From Time .....	439
<i>Carsten Baum, Bernardo David, Rafael Dowsley, Ravi Kishore, Jesper Buus Nielsen, and Sabine Oechsner</i>	
Efficient and Universally Composable Single Secret Leader Election from Pairings .....	471
<i>Dario Catalano, Dario Fiore, and Emanuele Giunta</i>	
Simple, Fast, Efficient, and Tightly-Secure Non-malleable Non-interactive Timed Commitments .....	500
<i>Peter Chvojka and Tibor Jager</i>	
Certifying Giant Nonprimes .....	530
<i>Charlotte Hoffmann, Pavel Hubáček, Chethan Kamath, and Krzysztof Pietrzak</i>	



Transparent Batchable Time-lock Puzzles and Applications to Byzantine Consensus .....	554
<i>Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri AravindaKrishnan Thyagarajan</i>	
<b>Pairings</b>	
Decentralized Multi-Authority Attribute-Based Inner-Product FE: Large Universe and Unbounded .....	587
<i>Pratish Datta and Tapas Pal</i>	
Multi-Client Inner Product Encryption: Function-Hiding Instantiations Without Random Oracles .....	622
<i>Elaine Shi and Nikhil Vanjani</i>	
GLUE: Generalizing Unbounded Attribute-Based Encryption for Flexible Efficiency Trade-Offs .....	652
<i>Marloes Venema and Greg Alpar</i>	
<b>Key Exchange and Messaging</b>	
EKE Meets Tight Security in the Universally Composable Framework .....	685
<i>Xiangyu Liu, Shengli Liu, Shuai Han, and Dawu Gu</i>	
A Universally Composable PAKE with Zero Communication Cost: (And Why It Shouldn't Be Considered UC-Secure) .....	714
<i>Lawrence Roy and Jiayu Xu</i>	
Sender-binding Key Encapsulation .....	744
<i>Laurin Benz, Wasilij Beskorovajnov, Sarai Eilebrecht, Jörn Müller-Quade, Astrid Ottenhues, and Rebecca Schwerdt</i>	
Pattern Matching in Encrypted Stream from Inner Product Encryption .....	774
<i>Élie Bouscatié, Guilhem Castagnos, and Olivier Sanders</i>	
<b>Author Index</b> .....	803

# **Homomorphic Cryptography and Other Topics**



# On Homomorphic Secret Sharing from Polynomial-Modulus LWE

Thomas Attema<sup>1,2,3</sup>, Pedro Capitão<sup>1,2(✉)</sup>, and Lisa Kohl<sup>1</sup>

<sup>1</sup> Cryptology Group, CWI, Amsterdam, The Netherlands  
`{pedro,lisa.kohl}@cwi.nl`

<sup>2</sup> Mathematical Institute, Leiden University, Leiden, The Netherlands

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

**Abstract.** Homomorphic secret sharing (HSS) is a form of secret sharing that supports the local evaluation of functions on the shares, with applications to multi-server private information retrieval, secure computation, and more.

Insisting on additive reconstruction, all known instantiations of HSS from “Learning with Error (LWE)”-type assumptions either have to rely on LWE with superpolynomial modulus, come with non-negligible error probability, and/or have to perform expensive ciphertext multiplications, resulting in bad concrete efficiency.

In this work, we present a new 2-party local share conversion procedure, which allows to *locally* convert noise encoded shares to non-noise plaintext shares such that the parties can detect whenever a (potential) error occurs and in that case resort to an alternative conversion procedure.

Building on this technique, we present the first HSS for branching programs from (Ring-)LWE with polynomial input share size which can make use of the efficient multiplication procedure of Boyle et al. (Eurocrypt 2019) and has no correctness error. Our construction comes at the cost of a – on expectation – slightly increased output share size (which is insignificant compared to the input share size) and a more involved reconstruction procedure.

More concretely, we show that in the setting of 2-server private information retrieval we can choose ciphertext sizes of only a quarter of the size of the scheme of Boyle et al. at essentially no extra cost.

## 1 Introduction

In 1979, Shamir introduced the concept of secret sharing information in his seminal paper *How to Share a Secret* [31]. In the two-party setting, secret sharing allows to split up a secret value into two secret shares, such that each share individually hides the secret, whereas the shares together allow to recover it. The simplest secret-sharing scheme is *additive secret sharing*, where a value  $x$  in an additive group  $\mathbb{G}$  is split into  $x_0, x_1$ , such that  $x_0, x_1$  are distributed uniformly at random conditioned on  $x_0 + x_1 = x$ . Despite its simplicity, additive secret

sharing comes with a number of nice properties. For example, it allows the *local* evaluation of linear functions on the shares.

In 2019, Boyle, Gilboa and Ishai [10] extended this notion to *homomorphic secret sharing (HSS)*, which allows the *local* evaluation of larger classes of function on the shares, while keeping the nice properties of additive secret sharing (so far possible). More precisely, a homomorphic secret-sharing scheme for a function class  $\mathcal{F}$  (over some input space  $\mathbb{G}$ ) has the following properties:

- The secret shares individually hide the message (*computationally*).
- The secret shares are succinct, i.e., they are *polynomial* in the size of the secret to be shared (in particular, they are independent of the complexity of the function class  $\mathcal{F}$ ).
- The secret shares allow local evaluation of all functions  $f \in \mathcal{F}$ . More precisely, there exists an evaluation procedure  $\text{Eval}$ , such that given secret shares  $x_0, x_1$  of  $x \in \mathbb{G}$ , it holds  $\text{Eval}(f, x_0) + \text{Eval}(f, x_1) = f(x)$ .

Note that the last condition explicitly requires *additive reconstruction*, i.e., evaluation results in an additive secret sharing of the output. While this requirement can be relaxed to more general reconstruction functions (as we will do in this work), it has a number of useful features, such as allowing the local postprocessing with linear functions.

Since their introduction, homomorphic secret sharing has found numerous applications, including 2-server private-information retrieval [9, 11, 19, 24, 32], low-communication secure computation [8, 10, 12, 20], and succinct generation of correlated (pseudo-)randomness [6, 7].

In [10], Boyle et al. presented a homomorphic secret-sharing scheme from the decisional Diffie-Hellman assumption for the class of *restricted multiplication straight-line (RMS) programs*. These programs are restricted in that they only allow multiplication between an input value and a memory value (where a memory value is an intermediate value in the computation), but not a multiplication between two memory values. It can be shown that this captures the class of polynomial-size branching programs, and circuits of constant fan-out and logarithmic depth (i.e., circuits in the complexity class  $\text{NC}^1$ ).

Since then, further HSS constructions for RMS programs have been proposed based on the decisional Diffie-Hellman assumption [8], the Paillier assumption [23, 28, 30], and based on the learning with errors (LWE) assumption [14, 16, 22]. All schemes, however, come with some efficiency bottleneck: either the evaluation is computationally expensive [8, 10, 16, 22, 23, 28, 30] and/or the input shares have high concrete overhead resulting in bad communication complexity [14, 16, 22].

In particular, while the scheme of Boyle et al. BKS [14] comes with desirable properties such as (plausible) post-quantum security and (comparatively) efficient multiplication on ciphertexts, it inherently has to rely on LWE with (double-)superpolynomial modulus (and thus large ciphertexts) in order to keep the error probability negligible. The reason for their (double-)superpolynomial modulus is a share conversion procedure to locally convert noise encoded shares modulo  $q$  to non-noisy shares modulo  $q$ . In order to achieve negligible error probability, they

need to choose moduli  $p, q$  with  $1 \ll p \ll q$ , where each  $\ll$  denotes a superpolynomial gap. The starting point for our work can thus be phrased as follows.

*Is it possible to design a share-conversion procedure for polynomial-sized  $p, q$  without introducing a non-negligible error?*

## 1.1 Our Contribution

In this paper, we answer this question (somewhat) affirmatively and present an HSS scheme from LWE for RMS programs with polynomial modulus, which otherwise inherits the nice properties from BKS. Our core technique is a share conversion which allows to locally detect and tentatively correct potential errors. On the downside, we have to relax additive reconstruction to a more involved reconstruction procedure, where the parties choose the output from an expected constant-size list of potential output values. In the following we give a high-level overview of our main results, which we discuss in more detail in the technical overview.

*Our Core Lemmas.* Our core technique can be captured in the following two lemmas for *share conversion*, a crucial step in the homomorphic evaluation of multiplications. Informally, the lemma states that (for rounding) there exist *local* conversion procedures that return shares  $\mathbf{flag}_0, z_0$  and  $\mathbf{flag}_1, z_1, z'_1$ , respectively, such that either  $z_0 = z_1 \pmod p$  or  $z_0 = z'_1 \pmod p$ , where the latter holds if and only if  $\mathbf{flag}_0 = \mathbf{flag}_1 = 1$ . This extends the technique of BKS, who only consider the case  $\mathbf{flag}_0 = \mathbf{flag}_1 = 0$  and choose parameters to ensure that this holds except with negligible probability.

**Lemma 1 (Rounding with correction [Lemmas 5, 6]).** *Let  $p, q \in \mathbb{N}$  with  $p|q$ . Then, there exist efficient procedures  $\text{Round}_0: \mathbb{Z}_q \rightarrow \{0, 1\} \times \mathbb{Z}_p$  and  $\text{Round}_1: \mathbb{Z}_q \rightarrow \{0, 1\} \times \mathbb{Z}_p^2$  such that the following holds:*

*For any  $x \in \mathbb{Z}_p$ , any  $e \in \mathbb{Z}$  with  $|e| < q/(4p)$ , and any  $t_0, t_1$  with*

$$t_0 + t_1 = \frac{q}{p} \cdot x + e \pmod q,$$

*it holds*

$$x = \begin{cases} z_0 + z_1 \pmod p & \text{if } \mathbf{flag}_0 = 0 \vee \mathbf{flag}_1 = 0, \\ z_0 + z'_1 \pmod p & \text{if } \mathbf{flag}_0 = \mathbf{flag}_1 = 1, \end{cases}$$

*where  $(\mathbf{flag}_0, z_0) \leftarrow \text{Round}_0(t_0)$  and  $(\mathbf{flag}_1, z_1, z'_1) \leftarrow \text{Round}_1(t_1)$ .*

*Further, for  $t_0, t_1$  chosen at random, it holds  $\mathbf{flag}_0 = \mathbf{flag}_1 = 0$  with probability at least  $1 - (4 \cdot |e| \cdot p)/q$ .*

Similarly, we extend their lemma for lifting.

**Lemma 2 (Lifting with correction [Lemmas 8, 9]).** *Let  $p, q \in \mathbb{N}$  with  $p|q$ . Then, there exist efficient procedures  $\text{Lift}_0: \mathbb{Z}_p \rightarrow \{0, 1\} \times \mathbb{Z}_q$  and  $\text{Lift}_1: \mathbb{Z}_p \rightarrow \{0, 1\} \times \mathbb{Z}_q^2$  such that the following holds:*

For any  $x \in \mathbb{Z}_p$ , with  $|x| < p/6$ , and any  $z_0, z_1$  with

$$z_0 + z_1 = x \pmod{p},$$

it holds

$$x = \begin{cases} v_0 + v_1 \pmod{q} & \text{if } \text{flag}_0 = 0 \vee \text{flag}_1 = 0, \\ v_0 + v'_1 \pmod{q} & \text{if } \text{flag}_0 = \text{flag}_1 = 1, \end{cases}$$

where  $(\text{flag}_0, v_0) \leftarrow \text{Lift}_0(z_0)$  and  $(\text{flag}_1, v_1, v'_1) \leftarrow \text{Lift}_1(z_1)$ .

Further, for  $z_0, z_1$  chosen at random it holds  $\text{flag}_0 = \text{flag}_1 = 0$  with probability at least  $1 - (4 \cdot |x|)/p$ .

*Our HSS.* We show that building on the core lemma, we obtain an HSS with one-sided error correction. More precisely,  $\mathcal{P}_0$  will follow a fixed computation path (remembering the wires where  $\text{flag}_0 = 1$ ). Party  $\mathcal{P}_1$  on the other hand, continues the computation for  $z_1$  and  $z'_1$  whenever  $\text{flag}_1 = 1$  for some wire. In the end, the parties can reconstruct the value by choosing the computation path that resorts to the alternative computation for  $\mathcal{P}_1$  whenever  $\text{flag}_0 = 1$  and  $\text{flag}_1 = 1$  for some wire. Note that this potentially results in *exponential* computation time for  $\mathcal{P}_1$ . We resolve this by choosing the parameters depending on the number of multiplications to be performed, such that the *overall* number of expected errors is 1 (or less). This means that on expectation  $\mathcal{P}_1$  has to perform the computation twice (from some point in the program on) and finally obtains two output shares. We want to stress that the output shares (corresponding to plaintext values) are typically several orders of magnitude smaller than the input shares (corresponding to ciphertext values). The increase in output values is therefore insignificant compared to the savings in input shares.

For instantiating our HSS, we present a trade-off between ciphertext size (equaling the input share size) and expected number of output shares. More precisely, instantiating the underlying public-key encryption scheme PKE with the Ring-LWE based encryption scheme of Lyubashevsky, Peikert and Regev [27] over the ring  $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ , we obtain the following.

**Lemma 3 (Corollary of Lemma 11).** *Let  $\gamma > 1$ . Let  $P$  be a branching program with multiplicative size  $|P|$  (i.e., number of load and multiplication operations) and magnitude bound  $B_{\max}$  (i.e., upper bound on all intermediary computation values). Then, setting  $p \geq 8 \cdot B_{\max} \cdot N \cdot |P| / \ln \gamma$  and  $q \geq 8 \cdot p \cdot N \cdot |P| / \ln \gamma$  in our HSS construction party  $\mathcal{P}_1$  obtains at most  $\gamma$  output shares on expectation.*

Setting  $\gamma = 1 + \lambda^{-\omega(\log \lambda)}$  (and thus obtaining  $1/\ln \gamma \approx \lambda^{\omega(\log \lambda)}$ ) we can recover the negligible error probability at the cost of superpolynomial ciphertext sizes of BKS.

*HSS with Perfect Correctness.* As a corollary of our techniques, we can obtain an HSS for RMS programs that satisfies *perfect* correctness, since the parties can *always* detect and correct the errors.

**Table 1.** Our HSS parameters for program size  $|P| = 2^{20}$ ,  $\gamma = 2$ .

$B_{\max}$	$N$	$\log q$
2	2048	71
$2^{16}$	2048	86
$2^{32}$	4096	104
$2^{64}$	4096	136
$2^{128}$	8192	202
$2^{256}$	8192	330

**Table 2.** BKS HSS parameters with *per gate* error probability  $2^{-40}$ .

$B_{\max}$	$N$	$\log q$
2	4096	137
$2^{16}$	4096	167
$2^{32}$	8192	203
$2^{64}$	8192	267
$2^{128}$	16384	399
$2^{256}$	16384	655

*Concrete Efficiency.* In Tables 1 and 2, we give concrete parameter sizes in comparison with the scheme of BKS, depending on the program size  $|P|$ . Note that the parameters of the BKS HSS scheme also have to grow with the program size of the underlying program  $|P|$  to ensure a fixed error probability, similarly to our scheme. Even without taking this into account (i.e., considering an error probability of  $2^{-40}$  after one operation rather than  $|P|$ ), it can be seen that our scheme can achieve a factor 4 shorter ciphertexts.

*HSS with Expected Constant-Time Evaluation.* The focus of our paper are applications where there is no privacy requirement for reconstruction, and thus *expected* constant-time evaluation can be dealt with by cutting off the computation after a fixed certain number of operations. We note though that the expected running time of the evaluation algorithms imposes challenges in applications such as secure two-party computation, where party  $\mathcal{P}_0$  can potentially derive information about the input from the response time of  $\mathcal{P}_1$ . We leave dealing with this issue as an interesting open question.

*Share Reconstruction with Privacy.* We note that (apart from the above described problem concerning run-time leakage) the problem of share reconstruction with privacy can be viewed as (one-server) private information retrieval by keywords [17] satisfying a strong notion of database privacy, where the client (here party  $\mathcal{P}_0$ ) is not allowed to learn anything about the number and content of the database held by the server (here party  $\mathcal{P}_1$ ), except for the queried entry. This can be viewed as a special case of labelled private-set intersection [15, 18] and can be instantiated by relying on somewhat homomorphic encryption. (Note here that the database for share reconstruction is very small on expectation, and thus even using expensive ciphertext multiplication for the final reconstruction would in typical applications not have a significant impact on the overall run time.)

*Impossibility of Fully Local Share Conversion.* To complement our result, we show that *no* direct local share conversion (i.e., not resorting to an alternative

conversion procedure) can achieve negligible error, showing that the BKS HSS scheme inherently requires either superpolynomial ciphertext or some postprocessing on the outputs.

*Limitation to 2-Party HSS.* As for BKS, our techniques are inherently limited to the two-party case, since we use some “symmetry” properties between the two shares. More precisely, we rely on the fact that if  $t_0 + t_1 = \frac{q}{p} \cdot x + e$ , then the distance of  $t_0$  and  $t_1$  to the next (potentially different) multiple of  $\frac{q}{p}$  differs only by  $|e|$ . This is no longer true for three or more parties, where local rounding results in a constant error probability (independent of  $p$  and  $q$ ). Going beyond the two-party case therefore inherently requires new techniques.

*Beyond HSS.* A corollary of our core lemma is that the secure reconstruction of  $x \bmod p$  given  $t_0 + t_1 = \frac{q}{p} \cdot x + e$  can be performed using a single string-OT, where party  $\mathcal{P}_0$  acts as the sender with input-bit  $\mathbf{flag}_0$  and  $\mathcal{P}_1$  acts as the receiver inputting  $(z_1, z_1)$  if  $\mathbf{flag}_1 = 0$  and  $(z_1, z'_1)$  else. This might have applications to encryption with 2-party distributed decryption, as used, e.g., in lattice-based electronic voting schemes.

*HSS Rounding vs. Learning with Rounding (LWR).* The rounding function which underlies [14] and this paper is essentially the same as the rounding function used for LWR [4]. While [4] uses non-distributed rounding to reduce the hardness of LWR to LWE (essentially building on the fact that the LWE error is “rounded away” with high probability), the line of work on constructing HSS via rounding needs a stronger property on distributed rounding towards achieving correctness. In particular, the techniques to reduce the modulus in the reduction from LWR to LWE from super-polynomial to polynomial [2, 5] do not appear to help in reducing the modulus for LWE-based HSS constructions.

## 1.2 Technical Overview

In the following, we give an overview of the idea behind our core lemma and our HSS construction. For the purpose of the technical overview, we assume  $\mathcal{R} = \mathbb{Z}$ ,  $n \in \mathbb{N}$ , and  $p = p(\lambda), q = q(\lambda) \in \mathbb{N}$  such that  $p|q$ . By writing  $p \ll q$ , we denote that  $q/p \in \lambda^{\omega(1)}$ .

*Restricted Multiplication Straight-Line Programs (RMS).* Recall that for RMS programs there is a distinction between *input values* (inputs to the program) and *memory values* (intermediary computation values) and the following operations are supported:

- Loading an input value into memory;
- Adding two memory values;
- Multiplying an input value with a memory value;
- Outputting a memory value.



The HSS Scheme of [14]. Our starting point is the HSS scheme of [14]. The basis of their construction is an encryption scheme with nearly linear encryption. More precisely, let  $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$  be a public-key encryption scheme over message space  $\mathbb{Z}_p$ , such that the secret key and ciphertext space is  $\mathbb{Z}_q^d$ . Recall that PKE satisfies *nearly linear decryption*, if for all secret keys  $\mathbf{s}$ , all messages  $m \in \mathbb{Z}_p$ , and all encryptions  $\mathbf{c}$  of  $m$ , it holds

$$\langle \mathbf{s}, \mathbf{c} \rangle \approx \frac{q}{p} \cdot m \pmod{q}.$$

Further, BKS requires that  $\mathbf{s}$  has only entries in  $\{-1, 0, 1\}$  (or otherwise small bounded values). As observed in [14], these requirements are indeed satisfied by (variants of) many lattice-based encryption schemes [3, 4, 25, 26, 29].

Now, if  $B_{\max} \in \mathbb{N}$  with  $B_{\max} \ll p \ll q/B_{\max}$ , then an HSS for RMS programs with magnitude bound  $B_{\max}$  can be obtained as follows.

**Key generation.** The HSS key generation generates a key pair according to the key generation algorithm  $\text{PKE.Enc}$  and outputs secret key shares  $\text{ek}_0 := \mathbf{s}_0$  to  $\mathcal{P}_0$  and  $\text{ek}_1 := \mathbf{s}_1$  to  $\mathcal{P}_1$ , s.t.,  $\mathbf{s}_0 + \mathbf{s}_1 = \mathbf{s}$  for the secret key  $\mathbf{s} \in \{0, 1\}^d$ .

**Input and memory values.** Values are stored as follows.

- **Input values:** Input values  $|x| \leq B$  are encrypted as  $\{\text{Enc}(x \cdot s_i)\}_{i \in [d]}$ , where  $s_i$  is the  $i$ -th component of  $\mathbf{s}$ . (Note that by the techniques of BKS this is possible given knowledge only of the public key of the underlying encryption scheme. We will give more details on this in the main body of the paper.)
- **Memory values:** Memory values  $|y| \leq B$  are secret shared as  $\mathbf{t}_0, \mathbf{t}_1$ , such that  $\mathbf{t}_0 + \mathbf{t}_1 = y \cdot \mathbf{s} \pmod{q}$ .

Note that adding two memory values is straightforward by the linearity of additive secret sharing. Further, assuming that the first component of the secret key  $\mathbf{s}$  is always one (which is straightforward to achieve), outputting a memory value mod  $q$  can be done by simply outputting the first entry of the corresponding share. Finally, loading an input value is equivalent to multiplying an input value by 1. We therefore restrict to describing the restricted multiplication in the following.

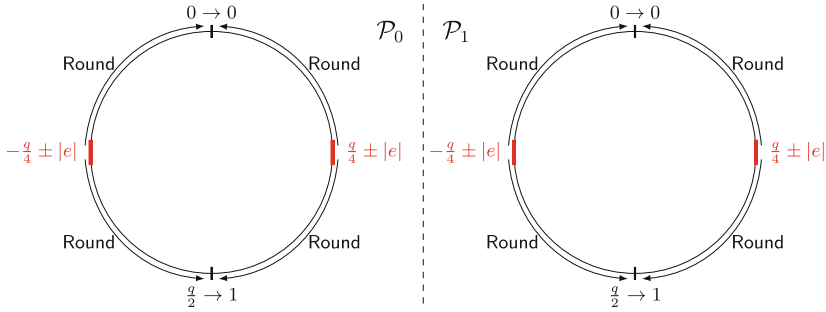
To perform a multiplication of an input value  $x$  encrypted as  $\{\mathbf{c}_i\}_{i \in [d]}$  with a memory value  $y$  shared as  $(\mathbf{t}_0, \mathbf{t}_1)$ , the idea is for the parties to locally compute  $\mathbf{t}_b^{\text{pre}}$  as  $t_{b,i}^{\text{pre}} := \langle \mathbf{c}_i, \mathbf{t}_b \rangle$ . By the property of nearly linear decryption, this yields:

$$t_{0,i}^{\text{pre}} + t_{1,i}^{\text{pre}} = \langle \mathbf{c}_i, y \cdot \mathbf{s} \rangle = y \cdot \langle \mathbf{c}_i, \mathbf{s} \rangle \approx \frac{q}{p} \cdot x \cdot y \cdot s_i \pmod{q},$$

and thus

$$\mathbf{t}_0^{\text{pre}} + \mathbf{t}_1^{\text{pre}} \approx \frac{q}{p} \cdot x \cdot y \cdot \mathbf{s} \pmod{q}.$$

The challenging part is to *locally* convert the shares  $\mathbf{t}_b^{\text{pre}}$  into memory values, i.e.,  $\mathbf{t}_0^{\text{out}} + \mathbf{t}_1^{\text{out}} = x \cdot y \cdot \mathbf{s} \pmod{q}$ . To that end, BKS [14] introduce the *rounding* and *lifting* technique, which allow local share conversion. In the following, we will focus on the *rounding* technique, since the *lifting* technique (to lift shares modulo  $p$  to shares modulo  $q$ ) can be adapted similarly.



**Fig. 1.** Depiction of the local rounding procedure. If both shares are *outside* the area highlighted in red, then no rounding error occurs. (Color figure online)

**Lemma 4 (Rounding [BKS [14]]).** *Let  $p, q \in \mathbb{N}$  such that  $p|q$ . Let  $x \in \mathbb{Z}_p$  and let  $e \in \mathbb{Z}$  with  $|e| \ll q/p$ . Let  $t_0, t_1 \in \mathbb{Z}_q$  be sampled uniformly at random subject to*

$$t_0 + t_1 = \frac{q}{p} \cdot x + e \pmod{q}.$$

*Then there exists an efficient deterministic procedure Round such that*

$$\text{Round}(t_0) + \text{Round}(t_1) = x \pmod{p}$$

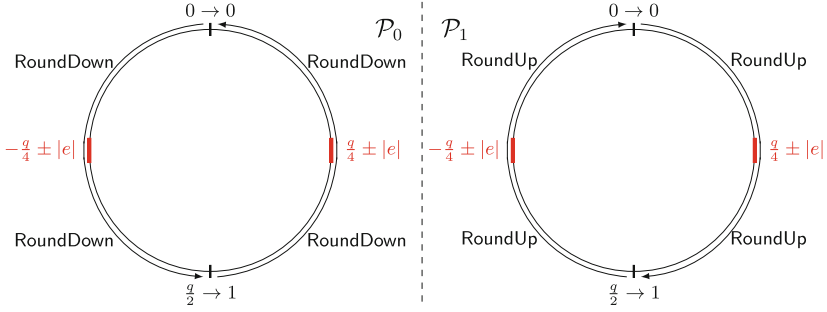
*except with negligible probability.*

*Towards HSS from Polynomial-Modulus LWE.* A straightforward approach towards HSS with polynomial modulus is to choose  $p, q$  of polynomial-size and handle the resulting non-negligible error with the generic error correction techniques of [10] introduced towards HSS from decisional Diffie-Hellman (where a non-negligible error is inherent [21]). These generic error correcting techniques come with a high concrete overhead though: If the error probability is a constant, then  $\omega(\log \lambda)$ -repetitions are necessary to achieve negligible error-probability via a majority vote. Thus, both the evaluation time and the size of the output shares are increased by a factor of  $\omega(\log \lambda)$ .

*This work: HSS from polynomial-modulus LWE with fine-grained error correction.* In this work, we show that in the case of LWE – and unlike decisional Diffie-Hellman – it is actually possible to detect (potential) errors, and therefore only correct if an error really occurs (or is very likely to occur). In order to outline our techniques, in the following we take a closer look at the rounding procedure from above.

To simplify presentation, for the rounding technique we assume  $p = 2$  and  $4|q$  (to ensure  $\frac{q}{2}$  and  $\frac{q}{4}$  are integers). We give a depiction of the rounding procedure in Fig. 1, where  $\text{Round}: \mathbb{Z}_q \rightarrow \mathbb{Z}_2$  is defined as

$$\text{Round}(y) := \left\lfloor \frac{2}{q} \cdot y \right\rfloor \pmod{2}.$$



**Fig. 2.** Depiction of the alternative local rounding procedure. If at least one of the shares is *inside* the area highlighted in red, then no rounding error occurs. (Color figure online)

Now, assume to be given shares  $t_0, t_1$  chosen at random conditioned on

$$t_0 + t_1 = \frac{q}{2} \cdot x + e,$$

where  $x \in \{0, 1\}$  and  $e$  is some error. Then, as observed in BKS [14], if at least one of the shares  $t_0, t_1$  is *outside* the *red area*  $[-\frac{q}{4} \pm |e|] \cup [\frac{q}{4} \pm |e|]$ ,<sup>1</sup> then no rounding error occurs, i.e.,

$$\left\lfloor \frac{2}{q} \cdot t_0 \right\rfloor + \left\lfloor \frac{2}{q} \cdot t_1 \right\rfloor = x \pmod{2}.$$

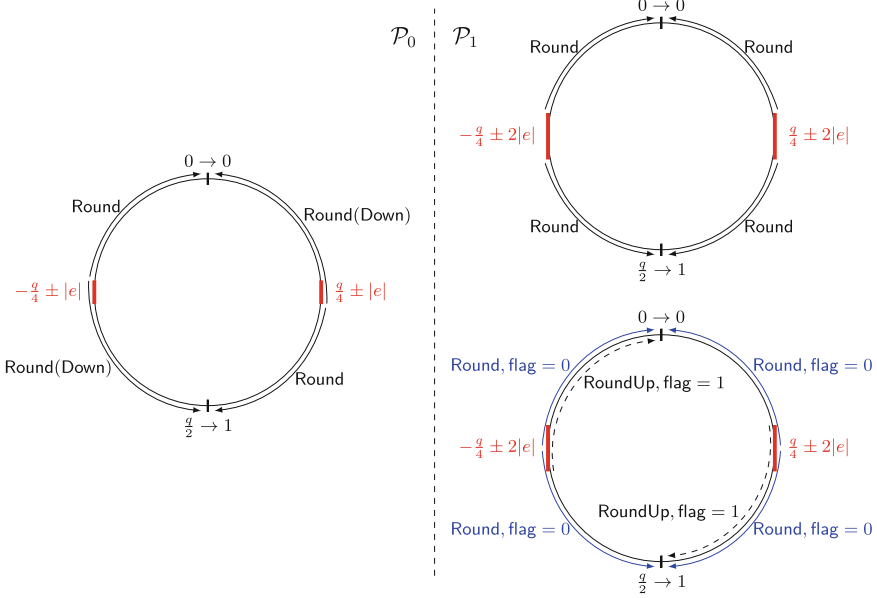
This crucially relies on the fact that for the shares it holds that  $t_0 + t_1 = e \pmod{q}$  or  $t_0 + t_1 = \frac{q}{2} + e \pmod{q}$ . Now, assume  $t_0$  is outside the red area and  $\text{Round}(t_0) = 0$  (the other cases are similar). Then, it must hold that  $t_0$  has distance  $< \frac{q}{4} - |e|$  from 0. Thus, if  $t_0 + t_1 = e$ , it must hold that  $t_1$  has distance  $< \frac{q}{4}$  from 0, and thus  $\text{Round}(t_1) = 0$  as required. On the other hand, if  $t_0 + t_1 = \frac{q}{2} + e \pmod{q}$ , then  $t_1$  must have distance  $< \frac{q}{4}$  from  $\frac{q}{2}$ , and thus  $\text{Round}(t_1) = 1$  as required.

If  $|e| \ll \frac{q}{2}$ , then the probability of a random element  $y \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  lying in the red area is negligible, and thus by the above considerations no rounding error occurs except with negligible probability.

Towards correcting the error, we observe that – on the other hand – if at least one of the shares  $t_0, t_1$  is *inside* one of the bad areas, then following an alternative procedure (depicted in Fig. 2) no rounding error occurs. The alternative rounding procedures  $\text{RoundDown}, \text{RoundUp}$  are defined as

$$\text{RoundDown}(x) := \left\lfloor \frac{2}{q} \cdot x \right\rfloor \pmod{2}, \quad \text{RoundUp}(x) := \left\lceil \frac{2}{q} \cdot x \right\rceil \pmod{2}.$$

<sup>1</sup> Here, we consider  $\mathbb{Z}_q$  to be represented as integers in the interval  $(-\frac{q}{2}, \frac{q}{2}]$ . For  $y \in \{-\frac{q}{4}, \frac{q}{4}\}$ , by  $[y \pm |e|]$  we denote the interval containing all  $z \in \mathbb{Z}_q$  having at most distance  $|e|$  from  $y$  (considered as integer).



**Fig. 3.** Depiction of the asymmetric local rounding procedure, where party  $\mathcal{P}_1$  is fully in charge of the error correction. (Color figure online)

In other words, party  $\mathcal{P}_0$  rounds all negative numbers  $(-\frac{q}{2}, -1]$  to  $-1 = 1 \pmod 2$ , and all positive number  $[1, \frac{q}{2}]$  to 0, and  $\mathcal{P}_1$  rounds all negative numbers  $(-\frac{q}{2}, -1]$  to 0, and all positive numbers  $[1, \frac{q}{2}]$  to 1 (and 0 is always rounded to 0).

The idea here is that if at least one of the shares  $t_0, t_1$  is *inside* the red area, then the other share is also  $|e|$ -close to the red area, and therefore one party rounding up and the other party rounding down always yields the correct result (as long as  $|e| < \frac{q}{4}$ ). More precisely, assume that  $t_0$  is in the red area and  $\text{Round}'(0, t_0) = 0$ , i.e.,  $t_0 \in [\frac{q}{4} \pm |e|]$  (the other cases are similar). Now, if  $t_0 + t_1 = e \pmod q$ , then  $t_1 \in [-\frac{q}{4} \pm 2 \cdot |e|]$ , and thus  $\text{Round}'(1, t_1) = 0$ . If  $t_0 + t_1 = \frac{q}{2} + e \pmod q$ , on the other hand, it holds  $t_1 \in [\frac{q}{4} \pm |e|]$  and thus  $\text{Round}'(1, t_1) = 1$  as required.

Given these two observations, we obtain our first core lemma (Lemma 1). We present the corresponding rounding procedures in Fig. 3. Here,  $\mathcal{P}_0$  always follows a fixed rounding procedure, where  $\mathcal{P}_0$  uses the normal rounding procedure *outside* the red area, and the rounding procedure  $\text{RoundDown}$  *inside* the red area. If its share is within the red area, it sets  $\text{flag} = 1$  for the corresponding wire, and  $\text{flag} = 0$  otherwise. If the share of  $\mathcal{P}_1$  is *outside* the (now larger) red area, it follows the standard rounding procedure, and sets  $\text{flag} = 0$ . If the share of  $\mathcal{P}_1$  is *inside* the larger red area, it follows both the standard rounding procedure (depicted by the blue arrows) and the  $\text{RoundUp}$  rounding procedure (depicted by the dashed arrows) and sets the flags to 0 and 1, respectively. For reconstruction,

the parties resort to the alternative (“dashed”) computation path whenever both parties set  $\text{flag} = 1$  on the corresponding wire.

Together with our new lifting lemma, this yields our HSS scheme. A crucial part of our construction is carefully taking account of the gates with  $\text{flag} = 1$ , which we explain in the main body.

## 2 Preliminaries

In this section we define the HSS primitive as well as the computational model for programs supported by our construction. We begin by introducing some notation. For  $n \in \mathbb{N}$ ,  $[n]$  denotes the set  $\{1, \dots, n\}$ . We denote by  $\lambda$  the security parameter.

We will work with the ring  $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ , where  $N \leq \text{poly}(\lambda)$  is a power of 2. The infinity norm on  $\mathcal{R}$  is defined as  $\|x\|_\infty = \max_{i \in [n]} |x_i|$  for  $x \in \mathcal{R}$  with coefficients  $x_1, \dots, x_n$ . For  $q \in \mathbb{N}$ , let  $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ , where we consider elements of  $\mathcal{R}_q$  to have all their coefficients in the interval  $(-q/2, \dots, q/2]$ .

### 2.1 Homomorphic Secret Sharing

We consider homomorphic secret sharing with a general decoding algorithm for the reconstruction of shares, as defined by Boyle et al. [13], in the public-key setting. We note that HSS is commonly defined with the stronger requirement of additive reconstruction, which enjoys several useful properties. By considering the more general definition, our scheme is able to forego some of those properties for efficiency. Moreover, we show that the decoding functionality can be easily and securely realized, depending on the application setting.

**Definition 1 (Homomorphic Secret Sharing).** *A 2-party public-key homomorphic secret sharing (HSS) scheme for a class of programs  $\mathcal{P}$  consists of algorithms (Gen, Enc, Eval, Dec) with the following syntax:*

- $\text{Gen}(1^\lambda)$  : *On input a security parameter  $1^\lambda$ , the key generation algorithm outputs a public key  $\text{pk}$  and a pair of evaluation keys  $(\text{ek}_0, \text{ek}_1)$ .*
- $\text{Enc}(\text{pk}, x)$  : *On input the public key  $\text{pk}$  and an input value  $x$ , the encryption algorithm outputs a ciphertext  $\mathbf{c}$ .*
- $\text{Eval}(\sigma, \text{ek}_\sigma, (\mathbf{c}_1, \dots, \mathbf{c}_n), P, \beta)$  : *On input a party index  $\sigma \in \{0, 1\}$ , evaluation key  $\text{ek}_\sigma$ , a vector of  $n$  ciphertexts, a program  $P \in \mathcal{P}$  with  $n$  input values, and an output modulus  $\beta$ , the homomorphic evaluation algorithm outputs a share  $y_\sigma$ .*
- $\text{Dec}(y_0, y_1, \beta)$  : *On input shares  $y_0, y_1$  and an output modulus  $\beta$ , the decoding algorithm outputs a value  $y$ .*

*The algorithms (Gen, Enc, Eval, Dec) should satisfy the following correctness and security requirements:*

**Perfect Correctness.** *For all  $\lambda \in \mathbb{N}$ , inputs  $x_1, \dots, x_n$ , program  $P \in \mathcal{P}$ , and integer  $\beta \geq 2$ , we have*

$$\text{Dec}(y_0, y_1, \beta) = P(x_1, \dots, x_n),$$

where  $(\text{pk}, \text{ek}_0, \text{ek}_1) \leftarrow \text{Gen}(1^\lambda)$ ,  $\mathbf{c}_i \leftarrow \text{Enc}(\text{pk}, x_i)$  for  $i \in [n]$  and  $y_\sigma \leftarrow \text{Eval}(\sigma, \text{ek}_\sigma, (\mathbf{c}_1, \dots, \mathbf{c}_n), P, \beta)$  for  $\sigma \in \{0, 1\}$ .

**Security.** For all  $\lambda \in \mathbb{N}$  and for all PPT adversaries  $\mathcal{A}$ ,

$$\Pr \left[ \mathcal{A}(\text{state}, \text{pk}, \text{ek}_\sigma, \mathbf{c}) = b \left| \begin{array}{l} (\sigma, x_0, x_1, \text{state}) \leftarrow \mathcal{A}(1^\lambda) \\ b \leftarrow \{0, 1\} \\ (\text{pk}, \text{ek}_0, \text{ek}_1) \leftarrow \text{Gen}(1^\lambda) \\ \mathbf{c} \leftarrow \text{Enc}(\text{pk}, x_b) \end{array} \right. \right] - \frac{1}{2} \leq \text{negl}(\lambda).$$

*Remark 1.* We relax the definition of HSS by not requiring the Eval algorithm to run in polynomial time, but only *expected* polynomial time, which will be the case in our construction. This can be converted into polynomial time by halting the computation after some fixed number of steps.

## 2.2 Restricted Multiplication Straight-Line Programs

Our HSS scheme supports homomorphic evaluation of the class of Restricted Multiplication Straight-line (RMS) programs. These are a restricted form of arithmetic circuits in which multiplication of intermediate values is not possible; only multiplication of an input value by an intermediate value (or *memory value*) is allowed.

**Definition 2 (RMS programs).** An RMS program over the ring  $\mathcal{R}$  consists of a magnitude bound  $B_{\max}$  and a sequence of instructions of the four types below, each indicating its ingoing and outgoing wires and ordered by a unique identifier  $\text{id} \in \mathbb{N}$ .

- Load input into memory: instruction  $(\text{load}, \text{id}, x, w)$  sets input  $x$  as a memory value in wire  $w$  ( $\hat{y}_w \leftarrow \hat{x}$ ).
- Add values in memory: instruction  $(\text{add}, \text{id}, u, v, w)$  adds the values in wires  $u$  and  $v$  ( $\hat{y}_w \leftarrow \hat{y}_u + \hat{y}_v$ ).<sup>2</sup>
- Multiply input by memory value: instruction  $(\text{mult}, \text{id}, x, v, w)$  multiplies the input  $x$  and the memory value in wire  $v$  ( $\hat{y}_w \leftarrow \hat{x} \cdot \hat{y}_v$ ).
- Output from memory: instruction  $(\text{out}, \text{id}, w)$  outputs the value in wire  $w$  as an element of  $\mathcal{R}_\beta$ .

If at any step of execution the magnitude of a memory value exceeds the bound  $B_{\max}$  (i.e.  $\|\hat{y}_w\|_\infty > B_{\max}$ ), the output of the program on the corresponding input is defined to be  $\perp$ . Otherwise the output is the sequence of values given by the out instruction.

We define the multiplicative size of an RMS program  $P$  as its number of load and mult instructions, and we denote it by  $|P|$ .

<sup>2</sup> We assume that for every instruction  $(\text{add}, \text{id}, u, v, w)$  such that  $u$  (resp.  $v$ ) is the output wire of a previous instruction with id  $\text{id}_u$  (resp.  $\text{id}_v$ ) we have  $\text{id}_u < \text{id}_v$ . This ensures that shares corresponding to  $u$  are computed before shares corresponding to  $v$  in our evaluation algorithm.

Note the distinction between the magnitude bound  $B_{\max}$  and the output modulus  $\beta$ . For example, in an RMS program computing a Boolean function  $f : \{0, 1\}^k \rightarrow \{0, 1\}$ , the input values 0 and 1 would be interpreted as integers,  $B_{\max}$  would be a bound on the greatest integer appearing as the result of an operation, and the output modulus would be  $\beta = 2$ . Our HSS scheme will require  $\beta \leq B_{\max} < p < q$ , where  $p$  and  $q$  are, respectively, the plaintext modulus and ciphertext modulus of the underlying encryption scheme.

*Remark 2* The definition of RMS program in [14] includes an additional operation type which allows input values to be added. The class of functions computable with this additional operation is the same, but it allows some functions to be computed using fewer multiplications, which may result in a more efficient homomorphic evaluation. We omit this operation from our definition, but we note that our HSS also supports it, in identical fashion to the BKS scheme. In both constructions this feature requires adjusting the bound on the ciphertext noise according to the maximum number of input additions, which influences the parameters of the scheme.

### 3 The Homomorphic Secret Sharing Scheme

In this section, we describe our homomorphic secret sharing scheme. Our HSS is an adaptation of the BKS scheme [14]. It supports homomorphic evaluations of the same class of functions: *Restricted Multiplication Straight-Line* (RMS) programs. Informally, we adapt the original BKS scheme by incorporating a new error reconciliation procedure. The protocol parameters of the BKS scheme are chosen such that correctness errors only occur with negligible probability. By contrast, our reconciliation procedure allows for smaller protocol parameters, since potential errors occurring during the homomorphic evaluations are corrected by the error reconciliation procedure. As a result the internal protocol parameters can be chosen to be polynomial in the security parameter, whereas BKS scheme requires superpolynomial protocol parameters, thereby reducing the communication complexity.

#### 3.1 The Protocol

Both the BKS scheme and our adaptation crucially rely on a public-key encryption scheme  $\text{PKE} = (\text{PKE.Gen}, \text{PKE.Enc}, \text{PKE.Dec})$  with *nearly linear decryption*, i.e., for all key-pairs  $(\text{pk}, \text{s}) \leftarrow \text{PKE.Gen}(1^\lambda)$ , messages  $m \in \mathbb{Z}_p$  and ciphertexts  $\mathbf{c} \leftarrow \text{PKE.Enc}_{\text{pk}}(m)$ , it holds that

$$\langle \mathbf{c}, \mathbf{s} \rangle = \frac{q}{p} \cdot m + e \pmod{q},$$

for some “small” noise term  $|e| \leq B_{\text{err}}$ .

HSS.Gen( $1^\lambda$ ): Generate  $(\mathbf{pk}, \mathbf{s}) \leftarrow \text{PKE.Gen}(1^\lambda)$  and sample a PRF key  $K \leftarrow \{0, 1\}^\lambda$  uniformly at random. Sample  $\mathbf{s}_0 \leftarrow \mathbb{Z}_q^d$  and define

$$\mathbf{s}_1 := \mathbf{s} - \mathbf{s}_0 \pmod q.$$

Output  $(\mathbf{pk}, \mathbf{ek}_0, \mathbf{ek}_1)$ , where  $\mathbf{ek}_0 := (K, \mathbf{s}_0)$ ,  $\mathbf{ek}_1 := (K, \mathbf{s}_1) \in \{0, 1\}^\lambda \times \mathbb{Z}_q^d$ .

**Fig. 4.** Homomorphic Secret Sharing - Key Generation.

Since the PKE has nearly linear decryption, the decryption procedure simply rounds the inner-product  $\langle \mathbf{c}, \mathbf{s} \rangle$  of the ciphertext and the secret key, multiplied by  $0 < p/q < 1$ , to the nearest integer, i.e.,

$$\text{PKE.Dec}(\mathbf{c}, \mathbf{s}) = \left\lceil \frac{p}{q} \cdot \langle \mathbf{c}, \mathbf{s} \rangle \right\rceil \pmod p.$$

We assume that the first coefficient of the secret key  $\mathbf{s} \in \mathbb{Z}^d$  equals 1. This property is crucially required by the HSS construction, and it is satisfied by most PKE schemes with nearly linear decryption.

Further, for simplicity, we assume PKE to be defined over  $\mathbb{Z}$ . For this reason, our homomorphic secret sharing scheme will also be defined over  $\mathbb{Z}$ . However, all techniques and results have a straightforward generalization to rings of the form  $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$  for  $N$  a power of 2, namely, the rounding and lifting procedures are applied to each of the  $N$  coordinates of elements of  $\mathcal{R}$ .

As shown in [14], if PKE has nearly linear decryption and pseudorandom ciphertexts, there exists a *Key Dependent Message* (KDM) oracle PKE.OKDM that, without knowledge of the secret key, outputs encryptions of scalar multiples of the secret key ([14], Lemma 3). More precisely, for all  $j \in \{1, \dots, d\}$  and  $x \in \mathbb{Z}$ ,

$$\mathbf{c}_j \leftarrow \text{PKE.OKDM}(\mathbf{pk}, x, j) \quad \text{s.t.} \quad \langle \mathbf{c}_j, \mathbf{s} \rangle = x \cdot s_j + e \pmod q,$$

where  $\mathbf{s} = (s_1, \dots, s_d)$  and  $|e| \leq B_{\text{err}}$ . By linearity, the KDM oracle allows encryptions of arbitrary linear combinations of the secret key to be generated.

Let us now continue to describe our 2-party homomorphic secret sharing scheme HSS. Besides a PKE scheme with the above properties, the HSS construction also requires a keyed pseudorandom function PRF. The key-generation of our HSS scheme, described in Fig. 4, is identical to that of the BKS scheme. The HSS public key is simply a public key for the PKE scheme and each evaluation key contains an additive secret share  $\mathbf{s}_\sigma$  of the secret key together with a PRF key  $K$ .

The second functionality of the HSS is encryption. It allows parties to encrypt the inputs to the RMS program that is to be evaluated. However, an HSS encryption of an input value  $x \in \mathbb{Z}$  is different from a standard PKE encryption of  $x$ . Instead, it is an encryption of the key-dependent vector  $x \cdot \mathbf{s} \in \mathbb{Z}_q^d$ , where  $\mathbf{s} \in \mathbb{Z}_q^d$  is the secret key corresponding to the public key  $\mathbf{pk}$  generated in the key generation. Hence, the HSS encryption of  $x$  is a vector of  $d$  PKE encryptions, each



HSS.Enc(pk,  $x$ ):    Compute  $\mathbf{c}_j \leftarrow \text{PKE.OKDM}(\text{pk}, x, j)$  for  $j = 1, \dots, d$ .  
 Output the ciphertext  $\mathbf{C} := (\mathbf{c}_1, \dots, \mathbf{c}_d)$ .

**Fig. 5.** Homomorphic Secret Sharing - Encryption.

to a different key-dependent message  $x \cdot s_i$  for  $i \in \{1, \dots, d\}$ . Note that, since  $\mathbf{s} = (s_1, s_2, \dots, s_d) \in \mathbb{Z}_q^d$ , the first component of an HSS encryption is a standard PKE encryption of  $x \cdot 1 = x$ . The HSS encryption functionality, again identical to the one used by the BKS scheme, is described in Fig. 5. Intuitively, security of our HSS scheme follows from the security of OKDM and from each share  $\mathbf{s}_\sigma$  individually hiding  $\mathbf{s}$ .

The reason for using this “key-dependent” encryption is that, by deploying a distributed decryption, the two parties can take encrypted input values and obtain additive secret shares of the vector  $x \cdot \mathbf{s}$ . The BKS scheme shows how to perform certain operations on secret shares of key-dependent messages of this form. More precisely, it shows that the following operations can be performed *locally* (i.e., without requiring interaction between the two parties):

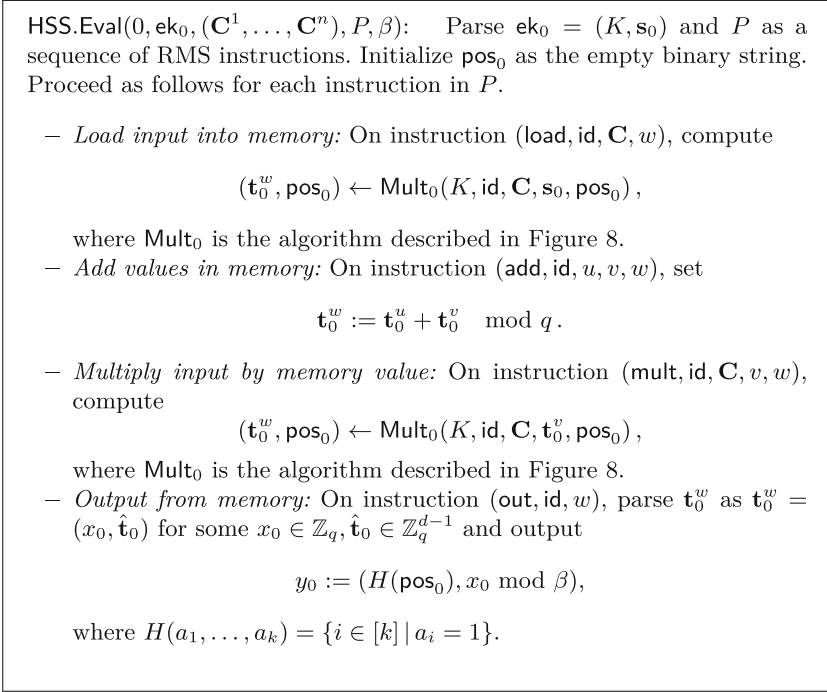
- **Addition:** given a secret share of  $x \cdot \mathbf{s}$  and a secret share  $y \cdot \mathbf{s}$ , obtain a secret share of  $(x + y) \cdot \mathbf{s}$ .
- **Multiplication by Input Value:** given an HSS encryption of  $x$  and a secret share of  $y \cdot \mathbf{s}$ , obtain a secret share of  $xy \cdot \mathbf{s}$ .

The HSS scheme thus distinguishes between (encrypted) input values and intermediate computation values, also referred to as *memory* values. The above functionalities immediately imply an HSS for RMS programs.

Our scheme deviates from BKS in how it performs the above HSS operations. In the BKS scheme these operations involve a distributed decryption, which in turn involves the rounding of a noisy value followed by a “lifting” of shares mod  $p$  to shares mod  $q$ . Both of these steps may fail, causing a correctness error, and the BKS scheme chooses its parameters such that such errors only occur with negligible probability. In our approach, we employ procedures **Round** and **Lift** (defined in Sect. 3.2) which indicate whether an error may have occurred and correct it if necessary.

In more detail, for party  $\mathcal{P}_0$ , the output of **Round** is of the form  $(\text{flag}_0, z_0) \in \{0, 1\} \times \mathbb{Z}_p$ . If  $\text{flag}_0 = 0$  (no error can occur), then  $z_0$  is obtained by rounding as usual, while if  $\text{flag}_0 = 1$  (an error may occur), then  $z_0$  is the result of an alternative “error-correcting” rounding.

Before describing the procedure for party  $\mathcal{P}_1$ , note that, since the parties cannot communicate, there is no guarantee that their flags will coincide. Moreover, the error-correcting requires the two parties to be in sync, i.e. correctness is not guaranteed if one party follows the usual rounding and the other the alternative rounding. Therefore, it may seem necessary that each party computes both the usual and alternative values when their flag is positive, in order to use one of them depending on the flag of the other party. However, we are able to define



**Fig. 6.** Homomorphic Secret Sharing - Evaluation for party  $\mathcal{P}_0$ .

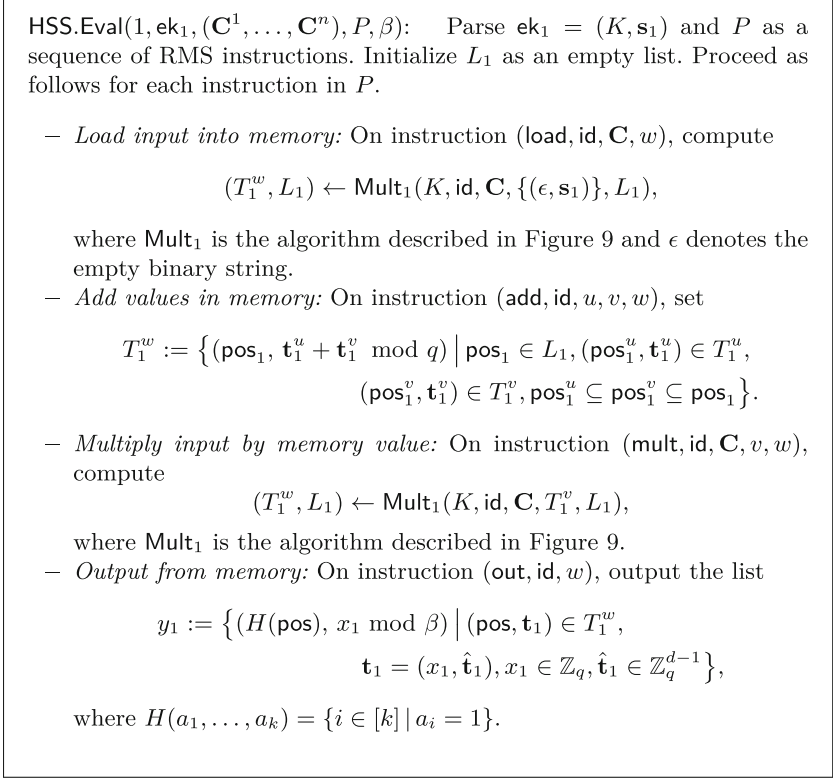
Round in a way such that whenever  $\text{flag}_0 = 1$  we have  $\text{flag}_1 = 1$  as well. This allows us to define Round for  $\mathcal{P}_0$  as described above, always computing a single value  $z_0$ , and have only  $\mathcal{P}_1$  compute two different values when  $\text{flag}_1 = 1$ .

For  $\mathcal{P}_1$ , Round either outputs  $\text{flag}_1 = 0$  and  $z_1$ , or  $\text{flag}_1 = 1$  and  $(z_1, z'_1)$ , where  $z_1$  and  $z'_1$  denote the outputs of the usual and alternative rounding, respectively. The following table displays the 3 different scenarios that may occur, and whether the parties should use corrected values or not.

		Flag of $\mathcal{P}_0$	
		0	1
Flag of $\mathcal{P}_1$	0	No Correction	—————
	1	No Correction	Error Correction

Similarly, errors can occur and be mitigated in the so-called lifting step, which always follows rounding.

The homomorphic evaluation procedure for party  $\mathcal{P}_0$  is presented in Fig. 6. For every wire  $w$  in the RMS program  $P$  we compute a vector  $\mathbf{t}_0^w \in \mathbb{Z}_q^d$  which is  $\mathcal{P}_0$ 's additive share of  $x^w \mathbf{s}$ , where  $x^w$  is the value of  $P$  at  $w$ . Throughout this algorithm we keep track of the variable  $\mathbf{pos}_0 \in \{0, 1\}^*$  which denotes the sequence of flags of  $\mathcal{P}_0$ . After each “multiplicative” operation (i.e. load or mult



**Fig. 7.** Homomorphic Secret Sharing - Evaluation for party  $\mathcal{P}_1$ .

instruction), the flags generated during that operation are appended to the string  $\text{pos}_0 \in \{0, 1\}^*$ . Adding a pseudorandom value  $\text{PRF}(K, \text{id})$  before each rounding step guarantees that the shares are always close to uniform, and therefore the occurrences of positive flags are independent from one instruction to another. Finally, the output of  $\text{Eval}$  consists of a compression  $H(\text{pos}_0)$  of the flag sequence of  $\mathcal{P}_0$  and the first component of  $\mathbf{t}_0^w$ , which is an additive share of  $P(x_1, \dots, x_n)$ . The compression function  $H$  simply outputs the list of indices with a flag set to 1 (which will be constant in number). The use of  $H$  is crucial in obtaining succinct output shares, as the size of  $\text{pos}_0$  is proportional to the size  $|P|$  of the program.

In Fig. 7 we present the homomorphic evaluation procedure for party  $\mathcal{P}_1$ , which is similar to that of  $\mathcal{P}_0$  but has an added degree of complexity, since  $\mathcal{P}_1$  generates two different possible values for its additive share whenever it gets a positive flag, and must keep track of all possible combinations. The global variable  $L_1$  in this algorithm is the list of binary strings which includes all possible sequences of flags of  $\mathcal{P}_0$  – recall that whenever  $\mathcal{P}_1$  has  $\text{flag}_1 = 0$  it knows that  $\text{flag}_0 = 0$ , but if  $\text{flag}_1 = 1$  then  $\text{flag}_0$  can be either 0 or 1. To each wire

```

Input  $(K, \text{id}, \mathbf{C}, \mathbf{t}, \text{pos})$ 

Parse  $\mathbf{C} = (\mathbf{c}_1, \dots, \mathbf{c}_d)$ 
For each  $i \in [d]$  :
     $(\text{flag}_i, z_i) \leftarrow \text{Round}(0, \langle \mathbf{t}, \mathbf{c}_i \rangle + \text{PRF}(K, (\text{id}, i)) \bmod q)$ 
     $(\text{flag}'_i, v_i) \leftarrow \text{Lift}(0, z_i)$ 
 $\mathbf{t}' \leftarrow (v_1, \dots, v_d)$ 
 $\text{pos}' \leftarrow \text{pos} \parallel \text{flag}_1 \parallel \text{flag}'_1 \parallel \dots \parallel \text{flag}_d \parallel \text{flag}'_d$ 

Output  $(\mathbf{t}', \text{pos}')$ 

```

**Fig. 8.** Algorithm  $\text{Mult}_0$ , employed by party  $\mathcal{P}_0$  on loading and multiplication instructions.

```

Input  $(K, \text{id}, \mathbf{C}, T, L)$ 

Parse  $\mathbf{C} = (\mathbf{c}_1, \dots, \mathbf{c}_d)$ ,  $T = ((\text{pos}_1, \mathbf{t}_1), \dots, (\text{pos}_\ell, \mathbf{t}_\ell))$ 
For each  $(i, j) \in [d] \times [\ell]$  :
     $V_{ij} \leftarrow \emptyset$ 
     $(\text{flag}_{ij}, z_{ij}^0, z_{ij}^1) \leftarrow \text{Round}(1, \langle \mathbf{t}_j, \mathbf{c}_i \rangle - \text{PRF}(K, (\text{id}, i)) \bmod q)$ 
     $(\text{flag}_{ij}^0, v_{ij}^{00}, v_{ij}^{01}) \leftarrow \text{Lift}(1, z_{ij}^0)$ 
     $V_{ij} \leftarrow V_{ij} \cup \{(00, v_{ij}^{00})\}$ 
    If  $\text{flag}_{ij}^0 = 1$ :
         $V_{ij} \leftarrow V_{ij} \cup \{(01, v_{ij}^{01})\}$ 
    If  $\text{flag}_{ij}^1 = 1$ :
         $(\text{flag}_{ij}^1, v_{ij}^{10}, v_{ij}^{11}) \leftarrow \text{Lift}(1, z_{ij}^1)$ 
         $V_{ij} \leftarrow V_{ij} \cup \{(10, v_{ij}^{10})\}$ 
        If  $\text{flag}_{ij}^1 = 1$ :
             $V_{ij} \leftarrow V_{ij} \cup \{(11, v_{ij}^{11})\}$ 
 $T' \leftarrow \{(\text{pos} \parallel a_1 \parallel \dots \parallel a_d, (v_1, \dots, v_d)) \mid j \in [\ell], \text{pos} \in L,$ 
     $\text{pos}_j \subseteq \text{pos}, (a_i, v_i) \in V_{ij}\}$ 

 $L' \leftarrow \{\text{pos} \mid (\text{pos}, \mathbf{t}) \in T'\}$ 

Output  $(T', L')$ 

```

**Fig. 9.** Algorithm  $\text{Mult}_1$ , employed by party  $\mathcal{P}_1$  on loading and multiplication instructions.

```

HSS.Dec( $y_0, y_1, \beta$ ): Parse the shares as  $y_0 = (u_0, x_0)$  and  $y_1 = \{(u_1^{(1)}, x_1^{(1)}), \dots, (u_1^{(k)}, x_1^{(k)})\}$ . Output  $x_0 + x_1^{(i)} \bmod \beta$ , where  $i$  is the unique index such that  $u_0 = u_1^{(i)}$ .

```

**Fig. 10.** Homomorphic Secret Sharing - Decoding.

$w$  in  $P$  we associate a list  $T_1^w$  of pairs of the form  $(\text{pos}_1, \mathbf{t}_1^w)$ , where  $\mathbf{t}_1^w$  is the additive share corresponding to the value of  $P$  at  $w$  and  $\text{pos}_1$  is the corresponding sequence of flags. The output of the evaluation algorithm for  $\mathcal{P}_1$  is a list of pairs of the same form as the output for  $\mathcal{P}_0$ , one for each possible flag sequence.

Finally, in the decoding algorithm, depicted in Fig. 10, we identify the additive shares  $x_0, x_1$  which correspond to the same sequence of flags and add them to obtain  $P(x_1, \dots, x_n)$ .

*Remark 3.* We omit an optimization step consisting of checking if the two values associated with a positive flag for  $\mathcal{P}_1$  are the same, which provides a reduction of the flag probability by a factor of 2 in both rounding and lifting.

*Remark 4.* Like the BKS scheme, our protocol can also be converted into a secret-key HSS version, which is more efficient for those applications which do not require the public-key capabilities.

### 3.2 Rounding and Lifting

Below we present our rounding procedure and analyse its properties. The corresponding step in the BKS protocol consists of multiplying the share  $v \in \mathbb{Z}_q$  by  $p/q$  and rounding it to the nearest integer to obtain a share in  $\mathbb{Z}_p$ . This introduces a correctness error with probability proportional to  $p/q$  (see Lemma 7). Our approach solves this issue by flagging instances in which an error could occur if both parties were to round their shares to the nearest integer and correcting it by having one party round up and the other round down in those instances.

Recall that we consider the representation  $\mathbb{Z}_n = \{-\lceil(n-1)/2\rceil, \dots, \lfloor(n-1)/2\rfloor\}$  for any  $n \in \mathbb{N}$ . We first define the operations `RoundDown`, `RoundUp` and `RoundNear`, which map a value  $v$  from  $\mathbb{Z}_q$  to  $\mathbb{Z}_p$  by scaling and then rounding it down, up, or to the nearest integer, respectively:

$$\begin{aligned} \text{RoundDown}(v) &= \lfloor (p/q) \cdot v \rfloor \pmod{p}, \\ \text{RoundUp}(v) &= \lceil (p/q) \cdot v \rceil \pmod{p}, \\ \text{RoundNear}(v) &= \lceil (p/q) \cdot v \rceil \pmod{p}. \end{aligned}$$

The deterministic procedure `Round`, which takes as input a party identifier  $\sigma \in \{0, 1\}$  and a value  $v \in \mathbb{Z}_q$ , is defined as follows:

$$\begin{aligned} \text{Round}(0, v) &= \begin{cases} (1, \text{RoundDown}(v)), & \text{if } v \in \text{bad}_{B_{\text{err}}}, \\ (0, \text{RoundNear}(v)), & \text{otherwise,} \end{cases} \\ \text{Round}(1, v) &= \begin{cases} (1, \text{RoundNear}(v), \text{RoundUp}(v)), & \text{if } v \in \text{bad}_{2B_{\text{err}}}, \\ (0, \text{RoundNear}(v), \perp), & \text{otherwise,} \end{cases} \end{aligned}$$

where  $\text{bad}_{B_{\text{err}}} = \{v \in \mathbb{Z}_q \mid |v \pmod{(q/p)}| \geq q/(2p) - B_{\text{err}}\}$  and  $\text{bad}_{2B_{\text{err}}}$  is analogously defined.

**Lemma 5 (Rounding correctness).** *Let  $p, q, B_{\text{err}} \in \mathbb{N}$  be such that  $q$  is a multiple of  $p$  and  $B_{\text{err}} < q/(4p)$ . Then, for any  $v_0, v_1 \in \mathbb{Z}_q$ ,  $m \in \mathbb{Z}_p$  and  $e \in \mathbb{Z}$  such that  $|e| \leq B_{\text{err}}$  and*

$$v_0 + v_1 = (q/p) \cdot m + e \pmod q,$$

*the outputs  $(\text{flag}_0, z_0) \leftarrow \text{Round}(0, v_0)$ ,  $(\text{flag}_1, z_1, z'_1) \leftarrow \text{Round}(1, v_1)$  satisfy the following:*

- (i) *If  $\text{flag}_0 = 0$ , then  $z_0 + z_1 = m \pmod p$ .*
- (ii) *If  $\text{flag}_0 = 1$ , then  $\text{flag}_1 = 1$  and  $z_0 + z'_1 = m \pmod p$ .*

*Proof.* Let  $v_0, v_1, m, e$  be such that  $v_0 + v_1 = (q/p) \cdot m + e \pmod q$  and  $|e| \leq B_{\text{err}}$ , and let  $(\text{flag}_0, z_0) \leftarrow \text{Round}(0, v_0)$ ,  $(\text{flag}_1, z_1, z'_1) \leftarrow \text{Round}(1, v_1)$ . To prove the first claim, assume that  $\text{flag}_0 = 0$ . Then there exist  $k, r \in \mathbb{Z}$  such that  $v_0 = (q/p) \cdot k + r$  and  $|r| < q/(2p) - B_{\text{err}}$ . Therefore

$$v_1 = (q/p) \cdot (m - k) + e - r \pmod q$$

and  $|e - r| \leq |e| + |r| < q/(2p)$ . It follows that

$$\begin{aligned} z_0 &= \lceil (p/q) \cdot v_0 \rceil = \lceil k + \underbrace{(p/q) \cdot r}_{\in(-1/2, 1/2)} \rceil = k \pmod p, \\ z_1 &= \lceil (p/q) \cdot v_1 \rceil = \lceil m - k + \underbrace{(p/q) \cdot (e - r)}_{\in(-1/2, 1/2)} \rceil = m - k \pmod p, \end{aligned}$$

which shows that  $z_0 + z_1 = m \pmod p$ .

We now prove the second claim. If  $\text{flag}_0 = 1$ , there exist  $k, r \in \mathbb{Z}$  such that  $v_0 = (q/p) \cdot k + r$  and  $q/(2p) - B_{\text{err}} \leq r \leq q/(2p) + B_{\text{err}}$ . The other share is then  $v_1 = (q/p) \cdot (m - k) + e - r \pmod q$ , where  $q/(2p) - 2B_{\text{err}} \leq e - r \leq q/(2p) + 2B_{\text{err}}$ , since  $|e| \leq B_{\text{err}}$ . Therefore  $|v_1 \pmod{(q/p)}| \geq q/(2p) - 2B_{\text{err}}$  and  $\text{flag}_1 = 1$ . Moreover, observe that  $e < r$  and  $(p/q) \cdot (r - e) < 1$ , since

$$r \leq q/(2p) + B_{\text{err}} < q/p - B_{\text{err}} \leq q/p + e.$$

It follows that

$$\begin{aligned} z_0 &= \lfloor (p/q) \cdot v_0 \rfloor = \lfloor k + \underbrace{(p/q) \cdot r}_{\in[0, 1]} \rfloor = k \pmod p, \\ z'_1 &= \lceil (p/q) \cdot v_1 \rceil = \lceil m - k + \underbrace{(p/q) \cdot (e - r)}_{\in(-1, 0]} \rceil = m - k \pmod p, \end{aligned}$$

and therefore  $z_0 + z'_1 = m \pmod p$ . □

**Lemma 6 (Rounding flag probability).** *Let  $p, q, B_{\text{err}} \in \mathbb{N}$  be such that  $q$  is a multiple of  $p$  and  $B_{\text{err}} < q/(4p)$ . Let  $v_0, v_1 \in \mathbb{Z}_q$  be uniformly random subject to*

$$v_0 + v_1 = (q/p) \cdot m + e \pmod q,$$

where  $m \in \mathbb{Z}_p$  and  $|e| \leq B_{\text{err}}$  are fixed. Let also  $(\text{flag}_1, z_1, z'_1) \leftarrow \text{Round}(1, v_1)$ . Then

$$\Pr[\text{flag}_1 = 1 \text{ and } z_1 \neq z'_1] = 2B_{\text{err}} \cdot (p/q).$$

*Proof.* Let  $u_1 = v_1 \bmod (q/p)$  and note that  $u_1$  is uniformly distributed in  $\mathbb{Z}_{q/p}$ . Recall that  $\text{flag}_1 = 1$  if and only if  $|u_1| \geq q/(2p) - 2B_{\text{err}}$ . Moreover,  $\text{RoundNear}(v_1) \neq \text{RoundUp}(v_1)$  if and only if the fractional part of  $(p/q) \cdot v_1$  is in the interval  $(0, 1/2)$ , which holds if and only if  $0 < u_1 < q/(2p)$ . Define the set

$$S = \{u \in \mathbb{Z}_{q/p} \mid q/(2p) - 2B_{\text{err}} \leq u < q/(2p)\}.$$

If  $q/p = 2k + 1$  for some  $k \in \mathbb{N}$ , then  $S = \{k - 2B_{\text{err}} + 1, \dots, k\}$ , while if  $q/p = 2k$  then  $S = \{k - 2B_{\text{err}}, \dots, k - 1\}$ . In both cases  $|S| = 2B_{\text{err}}$ . Therefore  $\Pr[\text{flag}_1 = 1 \text{ and } z_1 \neq z'_1] = \Pr[u_1 \in S] = |S| \cdot (p/q) = 2B_{\text{err}} \cdot (p/q)$ .  $\square$

**Lemma 7 (Rounding error probability).** *Let  $p, q, B_{\text{err}} \in \mathbb{N}$  be such that  $q$  is a multiple of  $p$  and  $B_{\text{err}} < q/(4p)$ . Let  $v_0, v_1 \in \mathbb{Z}_q$  be random subject to*

$$v_0 + v_1 = (q/p) \cdot m + e \pmod{q},$$

where  $m \in \mathbb{Z}_p$  and  $|e| \leq B_{\text{err}}$  are fixed. Then

$$\Pr[\text{RoundNear}(v_0) + \text{RoundNear}(v_1) \neq m \pmod{p}] \geq (|e| - 1) \cdot (p/q).$$

*Proof.* Define  $u_\sigma = v_\sigma \bmod (q/p)$ , for  $\sigma = 0, 1$ , and assume first that  $e < 0$ . Observe that, if  $u_0, u_1 \in (0, q/(2p))$ , then a rounding error occurs: since  $e = u_0 + u_1 \bmod (q/p)$  and  $-(q/p) < e < 0$ , it must be the case that  $e = u_0 + u_1 - (q/p)$ , and therefore  $\text{RoundNear}(v_0) + \text{RoundNear}(v_1) = m - 1$ . If  $q/p = 2k + 1$  for some  $k \in \mathbb{N}$ , then

$$\Pr[u_0, u_1 \in (0, q/(2p))] = \Pr[u_0 \in \{k + e + 1, \dots, k\}] = |e| \cdot (p/q).$$

Alternatively, if  $q/p = 2k$ , then

$$\Pr[u_0, u_1 \in (0, q/(2p))] = \Pr[u_0 \in \{k + e + 1, \dots, k - 1\}] = (|e| - 1) \cdot (p/q).$$

By a similar reasoning it can be seen that in the case  $e \geq 0$  a rounding error occurs with probability at least  $|e| \cdot (p/q)$ , if  $q/p$  is odd, or  $(|e| + 1) \cdot (p/q)$ , if  $q/p$  is even.  $\square$

Now we present the lifting procedure, which always follows rounding. In the BKS protocol this step is simply an inclusion: a share  $z \in \mathbb{Z}_q$  becomes  $z \in \mathbb{Z}_p$ . However, as shown in Lemma 10, a correctness error occurs with probability proportional to  $1/p$ . Again, our new procedure overcomes this issue by predicting and correcting possible errors to guarantee that additive shares modulo  $p$  are always converted into shares modulo  $q$  of the same secret value.

The deterministic procedure **Lift**, which takes as input a party identifier  $\sigma \in \{0, 1\}$  and a value  $z \in \mathbb{Z}_p$ , is defined as follows:

$$\text{Lift}(0, z) = \begin{cases} (1, z), & \text{if } z \in \text{bad}_{B_{\max}}^+, \\ (1, z + p), & \text{if } z \in \text{bad}_{B_{\max}}^-, \\ (0, z), & \text{otherwise,} \end{cases}$$

$$\text{Lift}(1, z) = \begin{cases} (1, z, z - p), & \text{if } z \in \text{bad}_{2B_{\max}}^+, \\ (1, z, z), & \text{if } z \in \text{bad}_{2B_{\max}}^-, \\ (0, z, \perp), & \text{otherwise,} \end{cases}$$

where  $\text{bad}_{B_{\max}}^+ = [p/2 - B, p/2)$ ,  $\text{bad}_{B_{\max}}^- = [-p/2, -p/2 + B]$ , and  $\text{bad}_{2B_{\max}}^+$ ,  $\text{bad}_{2B_{\max}}^-$  are analogously defined. The proofs of the following three lemmas can be found in the full version of this paper.

**Lemma 8 (Lifting correctness).** *Let  $p, B_{\max} \in \mathbb{N}$  be such that  $B_{\max} < p/6$ . Then, for any  $z_0, z_1 \in \mathbb{Z}_p$ ,  $m \in \mathbb{Z}$  such that  $|m| \leq B_{\max}$  and*

$$z_0 + z_1 = m \pmod{p},$$

*the outputs  $(\text{flag}_0, v_0) \leftarrow \text{Lift}(0, z_0)$ ,  $(\text{flag}_1, v_1, v'_1) \leftarrow \text{Lift}(1, z_1)$  satisfy the following:*

- (i) *If  $\text{flag}_0 = 0$ , then  $v_0 + v_1 = m$  over  $\mathbb{Z}$ .*
- (ii) *If  $\text{flag}_0 = 1$ , then  $\text{flag}_1 = 1$  and  $v_0 + v'_1 = m$  over  $\mathbb{Z}$ .*

**Lemma 9 (Lifting flag probability).** *Let  $p, B_{\max} \in \mathbb{N}$  be such that  $B_{\max} < p/6$ . Let  $z_0, z_1 \in \mathbb{Z}_p$  be random subject to*

$$z_0 + z_1 = m \pmod{p},$$

*where  $m \in \mathbb{Z}_p$ . Let also  $(\text{flag}_1, v_1, v'_1) \leftarrow \text{Lift}(1, z_1)$ . Then*

$$\Pr[\text{flag}_1 = 1 \text{ and } v_1 \neq v'_1] = 2B_{\max}/p.$$

**Lemma 10 (Lifting error probability).** *Let  $p, B_{\max} \in \mathbb{N}$  be such that  $B_{\max} < p/6$ . Let  $z_0, z_1 \in \mathbb{Z}_p$  be random subject to*

$$z_0 + z_1 = m \pmod{p},$$

*where  $m \in \mathbb{Z}_p$ . Then*

$$\Pr[z_0 + z_1 \neq m] \geq (|m| - 1)/p.$$

We can now prove our main result.

**Theorem 1. (HSS correctness and security).** *Let PKE be a public-key encryption scheme with plaintext space  $\mathcal{R}_p$  and ciphertext space  $\mathcal{R}_q^d$ , satisfying the properties of nearly linear decryption (with error bound  $B_{\text{err}}$ ) and pseudorandom ciphertexts, such that  $B_{\text{err}} < q/(4p)$ . Let also PRF be a pseudorandom function taking values in  $\mathcal{R}_q$ . Then the 2-party homomorphic secret sharing*



scheme described in Figs. 4, 5, 6, 7, 8, 9 and 10 is perfectly correct and secure, as per Definition 1, and supports homomorphic evaluation of polynomial-sized RMS programs with magnitude bound  $B_{\max}$  and output modulus  $\beta$  such that  $\beta \leq B_{\max} < p/6$ .

*Proof.* Security follows immediately from the security of the BKS HSS [14], as the algorithms Gen and Enc are identical in the two schemes and the security definition is independent of the Eval algorithm. Note that this is a consequence of KDM security and of the fact that the evaluation keys individually hide the secret encryption key.

We will now show that our scheme satisfies perfect correctness. Let  $y_0 = (H(\mathbf{pos}_0), z_0)$  and  $y_1 = \{(H(\mathbf{pos}_1^{(1)}), z_1^{(1)}), \dots, (H(\mathbf{pos}_1^{(k)}), z_1^{(k)})\}$  be the evaluated shares corresponding to an RMS program  $P$  on input  $x_1, \dots, x_n$ .

Observe that, according to the definition of the algorithms  $\text{Mult}_0$  and  $\text{Mult}_1$ , there always exists  $i^* \in [k]$  such that  $\mathbf{pos}_1^{(i^*)} = \mathbf{pos}_0$ . This follows from the fact that, at any rounding or lifting step with position tag  $\mathbf{pos}$ , party  $\mathcal{P}_1$  always computes a value associated to  $\mathbf{pos}||0$  and, by part (ii) of Lemmas 5 and 8, if  $\mathcal{P}_0$  has a value associated to  $\mathbf{pos}||1$  then so does  $\mathcal{P}_1$ . Furthermore, the index  $i^*$  is unique, since the binary strings  $\mathbf{pos}_1^{(j)}$  are all distinct. Since the compression function  $H$  is injective, the only index  $i$  such that  $H(\mathbf{pos}_1^{(i)}) = H(\mathbf{pos}_0)$  is  $i^*$ .

We will show below that, during homomorphic evaluation of  $P$ , for all wires  $w$  we have

$$\mathbf{t}_0^w + \mathbf{t}_1^w = x^w \mathbf{s} \pmod{q} \quad (1)$$

whenever  $(\mathbf{pos}_1, \mathbf{t}_1^w) \in T_1^w$  and  $\mathbf{pos}_1^w = \mathbf{pos}_0^w$ , where  $\mathbf{pos}_0^w$  is the flag sequence  $\mathbf{pos}_0$  of  $\mathcal{P}_0$  at the time wire  $w$  is evaluated,  $x^w \in \mathcal{R}$  denotes the value of  $P$  at  $w$  and  $\mathbf{s} = (1, \hat{\mathbf{s}}) \in \mathcal{R} \times \mathcal{R}^{d-1}$  is the PKE secret key.

The final output will be  $\text{Dec}(y_0, y_1, \beta) = z_0 + z_1^{(i^*)} \pmod{\beta}$ , where  $(z_0, \hat{\mathbf{t}}_0) = \mathbf{t}_0^w$ ,  $(z_1^{(i^*)}, \hat{\mathbf{t}}_1) = \mathbf{t}_1^w$ ,  $(\mathbf{pos}_1^{(i^*)}, \mathbf{t}_1^w) \in T_1^w$  for an output wire  $w$  and  $\mathbf{pos}_1^{(i^*)} = \mathbf{pos}_0$ . If Eq. (1) holds, then by looking only at the first component of each vector in the equation we see

$$z_0 + z_1^{(i^*)} = x^w \cdot 1 = P(x_1, \dots, x_n) \pmod{q},$$

hence  $\text{Dec}(y_0, y_1, \beta) = P(x_1, \dots, x_n)$  with probability 1.<sup>3</sup>

It remains only to check that Eq. (1) holds for every instruction in  $P$  of type load, add or mult.

- For instruction  $(\text{load}, \text{id}, (\mathbf{c}_1, \dots, \mathbf{c}_d), w)$ , where  $\mathbf{c}_i \leftarrow \text{PKE.OKDM}(\text{pk}, y, i)$ , by the nearly linear decryption property we have

$$\begin{aligned} (\mathbf{t}_0^w)_i + (\mathbf{t}_1^w)_i &= \langle \mathbf{s}_0, \mathbf{c}_i \rangle + \text{PRF}(K, (\text{id}, i)) + \langle \mathbf{s}_1, \mathbf{c}_j \rangle - \text{PRF}(K, (\text{id}, i)) \\ &= \langle \mathbf{s}, \mathbf{c}_i \rangle = (q/p) \cdot y \cdot s_i + e_i \pmod{q} \end{aligned}$$

<sup>3</sup> We assume here that  $\beta$  divides  $q$ , so that shares mod  $q$  are also shares mod  $\beta$ . If we wish to avoid this assumption, we can simply perform a lifting step to obtain shares over  $\mathbb{Z}$  before reducing them mod  $\beta$ .

for some  $|e_i| \leq B_{\text{err}}$ .<sup>4</sup> We can thus apply Lemma 5 followed by Lemma 8 to conclude that, for the matching flags (i.e.  $\text{pos}_1^w = \text{pos}_0^w$ ), the corresponding shares  $\mathbf{t}_0^w, \mathbf{t}_1^w$  satisfy  $\mathbf{t}_0^w + \mathbf{t}_1^w = y\mathbf{s} \pmod q$ .

- For instruction  $(\text{add}, \text{id}, u, v, w)$ , assume Eq. (1) holds for  $(\mathbf{t}_0^u, \mathbf{t}_1^u)$  and  $(\mathbf{t}_0^v, \mathbf{t}_1^v)$ , where  $\text{pos}_0^u \subseteq \text{pos}_1^v \subseteq \text{pos}_1^w$  and  $\text{pos}_0^\tau = \text{pos}_1^\tau$  for  $\tau \in \{u, v, w\}$ . Then

$$\mathbf{t}_0^w + \mathbf{t}_1^w = \mathbf{t}_0^u + \mathbf{t}_0^v + \mathbf{t}_1^u + \mathbf{t}_1^v = x^u\mathbf{s} + x^v\mathbf{s} = x^w\mathbf{s} \pmod q.$$

- For instruction  $(\text{mult}, \text{id}, (\mathbf{c}_1, \dots, \mathbf{c}_d), v, w)$ , assuming Eq. (1) holds for  $(\mathbf{t}_0^v, \mathbf{t}_1^v)$  we have

$$\begin{aligned} (\mathbf{t}_0^w)_i + (\mathbf{t}_1^w)_i &= \langle \mathbf{t}_0^v, \mathbf{c}_i \rangle + \text{PRF}(K, (\text{id}, i)) + \langle \mathbf{t}_1^v, \mathbf{c}_j \rangle - \text{PRF}(K, (\text{id}, i)) \\ &= x^v \langle \mathbf{s}, \mathbf{c}_i \rangle = (q/p)x^v \cdot y \cdot s_i + e_i \pmod q \end{aligned}$$

and as in the load instruction we conclude that  $\mathbf{t}_0^w + \mathbf{t}_1^w = x^v y\mathbf{s} \pmod q$ .

□

### 3.3 Impossibility of Local Share Conversion

The next theorem shows that the local share conversion procedure that lies at the heart of lattice-based HSS cannot achieve perfect correctness with additive reconstruction. Therefore one must either allow correctness error (which can only be made negligible with a superpolynomial modulus) or relax the requirement for reconstruction.

**Theorem 2 (Correctness error of share conversion).** *Let  $m \in \mathbb{Z}_p$  and  $e \in D$ , where  $\{0, 1, -1\} \subseteq D \subseteq (-q/(2p), q/(2p))$ . Let also  $v_0, v_1 \in \mathbb{Z}_q$  be sampled uniformly subject to*

$$v_0 + v_1 = (q/p) \cdot m + e \pmod q.$$

*Then, for any local share conversion functions  $g_0, g_1 : \mathbb{Z}_q \rightarrow \mathbb{Z}_q$ , there exist  $m \in \mathbb{Z}_p$  and  $e \in D$  such that*

$$\Pr[g_0(v_0) + g_1(v_1) \neq m \pmod q] \geq p/(3q).$$

*Proof.* We show that in each interval  $I_k \subseteq \mathbb{Z}_q$  of the form  $I_k := [k \cdot q/p, (k+1) \cdot q/p)$  there exists  $v_0 \in I_k$  such that an error  $g_0(v_0) + g_1(v_1) \neq m$  occurs for at least one of the pairs  $(m, e) := (0, 0)$ ,  $(m, e) := (1, -1)$  or  $(m, e) := (0, 1)$ . Since there are  $p$  disjoint intervals  $I_k$ , one of these three choices of  $(m, e)$  must have at least  $p/3$  values  $v_0$  in the above conditions and the result follows from the fact that  $v_0$  is uniform.

To prove the above claim, consider  $v_0 := k \cdot q/p$  and  $v_1 := -v_0$ . If  $g_0(v_0) + g_1(v_1) \neq 0$  we have found an error for  $(m, e) := (0, 0)$ , as  $v_0 + v_1 = 0$  and  $v_0 \in I_k$ . Meanwhile,  $v'_0 := (k+1) \cdot q/p - 1$  satisfies  $v'_0 + v_1 = q/p \cdot 1 - 1$ , hence

<sup>4</sup> Here we again consider the case  $\mathcal{R} = \mathbb{Z}$  for simplicity. For  $\mathcal{R}$  of dimension  $N$ , the equation applies to each coordinate of  $y$ .

if  $g_0(v'_0) + g_1(v_1) \neq 1$  we have found an error for  $(m, e) := (1, -1)$ . Suppose now that  $g_0(v_0) + g_1(v_1) = 0$  and  $g_0(v'_0) + g_1(v_1) = 1$ . Then  $g_0(v_0) \neq g_0(v'_0)$  and there must exist  $\tilde{v}_0 \in [v_0, v'_0)$  such that  $g_0(\tilde{v}_0) \neq g_0(\tilde{v}_0 + 1)$ . Note that  $\tilde{v}_0, \tilde{v}_0 + 1 \in I_k$ . Then, unless an error occurs with  $\tilde{v}_0$  and  $(m, e) := (0, 0)$  or  $\tilde{v}_0 + 1$  and  $(m, e) := (0, 1)$ , by taking  $\tilde{v}_1 := -\tilde{v}_0$  we obtain

$$g_0(\tilde{v}_0) + g_1(\tilde{v}_1) = g_0(\tilde{v}_0 + 1) + g_1(\tilde{v}_1) = 0,$$

since  $\tilde{v}_0 + \tilde{v}_1 = 0$  and  $(\tilde{v}_0 + 1) + \tilde{v}_1 = 1$ . This contradicts the assumption  $g_0(\tilde{v}_0) \neq g_0(\tilde{v}_0 + 1)$ .  $\square$

## 4 Efficiency and Parameters

In this section we compute concrete parameters for our HSS scheme and compare them with the BKS scheme [14]. The next lemma gives us an expression for the average number of elements of the list that constitutes the share  $y_1$  of party  $\mathcal{P}_1$  after evaluating a program  $P$ . We are then able to choose parameters such that this number is bounded by a constant. Since the running time of the evaluation algorithm of  $\mathcal{P}_1$  is proportional to this quantity, the lemma also implies that it runs in expected polynomial time.

**Lemma 11 (Expected share size).** *Consider the HSS scheme described above, with ciphertext space  $\mathcal{R}_q^d$ , where  $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ . Let  $P$  be an RMS program of multiplicative size  $|P|$ . Denote by  $p_{\text{round}}$ ,  $p_{\text{lift}}$  the probabilities of party  $\mathcal{P}_1$  having a positive flag in a single rounding or lifting step, respectively. Then the expected total number  $E$  of terminal values in the homomorphic evaluation of  $P$  by  $\mathcal{P}_1$  is*

$$E = ((1 + p_{\text{round}})(1 + p_{\text{lift}}))^{dN|P|}.$$

We defer the proof of Lemma 11 to the full version. As a consequence of Lemmas 6, 9 and 11, we obtain the following bound, which we can use to choose parameters for the HSS scheme:

$$E \leq ((1 + 2B_{\text{err}}p/q)(1 + 2B_{\text{max}}/p))^{dN|P|}.$$

We instantiate PKE with the Ring-LWE based encryption scheme of Lyubashevsky, Peikert and Regev [27] over the ring  $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ , giving us  $B_{\text{err}} = 1$ ,  $d = 2$ . Then, if we wish to bound the expected number of terminal values  $E$  by some value  $\gamma > 1$ , setting  $p \geq 8B_{\text{max}}N|P|/\ln \gamma$  and  $q \geq 8pN|P|/\ln \gamma$  gives

$$E \leq (1 + \ln \gamma / (4N|P|))^{4N|P|} \leq \gamma,$$

which justifies that  $\gamma$  is indeed an upper bound. For instance, if we choose  $\gamma = 2$ , party  $\mathcal{P}_1$  will have, on average, a single positive flag throughout the homomorphic evaluation and two terminal values on which to perform reconstruction.

**Table 3.** HSS parameters for  $|P| = 2^{10}$ ,  $\gamma = 2$ .

$B_{\max}$	$N$	$\log q$	Security
2	2048	51	147.3
$2^{16}$	2048	66	109.4
$2^{32}$	2048	82	86.0
$2^{64}$	4096	116	122.9
$2^{128}$	8192	182	159.5
$2^{256}$	8192	310	89.1

**Table 4.** HSS parameters for  $|P| = 2^{20}$ ,  $\gamma = 2$ .

$B_{\max}$	$N$	$\log q$	Security
2	2048	71	100.9
$2^{16}$	2048	86	81.6
$2^{32}$	4096	104	139.0
$2^{64}$	4096	136	103.0
$2^{128}$	8192	202	141.7
$2^{256}$	8192	330	83.6

**Table 5.** BKS HSS parameters, with error probability  $2^{-40}$ .

$B_{\max}$	$N$	$\log q$	Security
2	4096	137	103.3
$2^{16}$	4096	167	83.7
$2^{32}$	8192	203	142.0
$2^{64}$	8192	267	104.9
$2^{128}$	16384	399	143.9
$2^{256}$	16384	655	84.6

In Tables 3 and 4 we present parameters of our scheme in this RLWE instantiation, namely the ring dimension  $N$  and the ciphertext modulus  $q$ , when we choose the bound  $\gamma = 2$  for the expected number of terminal values and maximum program sizes  $2^{10}$  and  $2^{20}$ , respectively. These are given in function of the magnitude bound  $B_{\max}$  of plaintexts during the computation. For comparison, Table 5 shows the parameters for the corresponding instantiation of the BKS HSS scheme. We observe that our scheme reduces the size of the modulus  $q$  by nearly a factor of 2 for programs with up to  $2^{20}$  operations (or by a greater factor, if we further restrict the program size) while also reducing  $N$  by a factor of 2 and attaining similarly high estimated computational security.

The security estimates on Tables 3, 4 and 5 were obtained by computing, for magnitude bound  $B_{\max}$ , the smallest pair  $(N, q)$  with at least 80 bits of computational security, as predicted by the lattice estimator tool of Albrecht et al. [1]. Note that the parameters of the BKS HSS scheme are also dependent on the size of the program  $P$ . The parameters on Table 5 correspond to a correctness error probability of  $2^{-40}$  for *each* (multiplicative) operation in  $P$ .

The parameter  $\gamma$  can be adjusted to reduce the frequency of raised flags for a relatively small cost in the size of lattice parameters. For instance, setting  $\gamma = 1.01$  boosts the probability that there are no raised flags in the entire computation to at least  $1 - (\gamma - 1) = 0.99$ , at the price of increasing the modulus  $q$  by a factor of  $(\ln 2 / \ln 1.01)^2 \approx 2^{12}$ , compared to the choice  $\gamma = 2$ .

On the other hand, since the size of the input shares is much larger than the size of the output shares, it can make sense to choose larger parameter  $\gamma$  for certain applications. One should note though, that if the parties wish to execute linear postprocessing on the output shares (e.g., a counting query over a large database), then the total expected number of shares scales with  $2^\gamma$ .

## 5 Applications

Our scheme retains most of the standard applications of HSS, even without having the usual property of additive reconstruction. It is particularly suited for scenarios where there is asymmetry between the parties performing the computation (e.g. two servers of different sizes).

*Private Database Queries.* We explore in detail one of the applications of the BKS HSS scheme and show that our construction provides an overall improvement in efficiency. A 2-server private database query protocol involves two non-colluding servers, each holding a copy of a public database DB, and a client, who can issue queries on the database. The protocol should allow the client to obtain the answer of its query while hiding both the query and the answer from the servers. HSS gives a simple solution to this problem with only one round of communication: in this protocol the client sends an encryption of its query to both servers, who then homomorphically compute shares of the answer and return them to the client. HSS for branching programs supports many expressive queries, such as conjunctive keyword search and pattern matching.

*Remark 5.* Unlike with secure 2-party computation, in this setting there are no concerns with the security of the reconstruction procedure. We can simply have both servers send their shares to the client, who evaluates the decoding algorithm directly with minimal computational cost.

*Linear Post-processing of Shares.* There are scenarios in which the additive reconstruction property of other HSS schemes is quite useful, such as when computing a counting query. This type of query returns the number of elements of the database satisfying some predicate  $Q$ , which can be written as  $\sum_{x \in \text{DB}} Q(x)$ , where  $Q(x) = 1$  if  $x$  satisfies  $Q$  and  $Q(x) = 0$  otherwise. Because of this additive representation, instead of homomorphically evaluating the query on the database at once, the servers can evaluate the predicate individually on each database element. The shares  $q_\sigma^x$  corresponding to each element  $x$  can then be locally summed to obtain  $q_\sigma = \sum_{x \in \text{DB}} q_\sigma^x$  and this value sent to the client, who recovers the result of the query as  $q_0 + q_1 \pmod{\beta}$ . In the BKS HSS scheme, this approach allows using the optimal case of  $B_{\max} = 2$  on the individual HSS evaluations, even though the query output is not bounded by 2.

Although our construction does not benefit from the additive reconstruction property, we can employ a similar technique and show that even in this setting we obtain a performance improvement. Recall that, in our scheme, a share evaluated by party  $\mathcal{P}_0$  is of the form  $(u, q)$  where  $u$  is the compression of the flag sequence of  $\mathcal{P}_0$  and  $q$  is the additive share of the result, while a share evaluated by  $\mathcal{P}_1$  is a list of pairs of the same form.  $\mathcal{P}_0$  can homomorphically evaluate  $Q$  on each  $x \in \text{DB}$  to obtain  $(u_0^x, q_0^x)$  and then send  $y_0 = (u_0^{x_1}, \dots, u_0^{x_M}, q_0 := \sum_{x \in \text{DB}} q_0^x)$  as its final share to the client, where  $M = |\text{DB}|$  is the database size. Similarly,  $\mathcal{P}_1$  obtains  $M$  lists from evaluating  $Q$  on every database element and its output to the client is a list of shares of the same form as  $y_0$ , one for each possible choice of a single element from each of the  $M$  lists. The client can then reconstruct by summing  $q_0$  and the corresponding value from  $\mathcal{P}_1$ 's share. This solution may look terribly inefficient for the fact that the size of  $\mathcal{P}_1$ 's output is proportional to the product of the number of elements of all  $M$  lists, but we can set the probability of any list having more than one element to be very low.

*A Concrete Example.* Consider a database DB with entries of the form  $(x, W_x)$  where  $x$  is a document and  $W_x$  is a list of keywords. Given a target list of key-

words  $W$ , we wish to count the number of documents containing all the keywords in  $W$ . That is, we consider a counting query for the predicate  $Q_W(x, W_x) = 1$  if  $W \subseteq W_x$ . Suppose the database size is  $M = 1024$ , the client's query consists of 4 keywords, and each document has 10 keywords with 128 bits of length. This can be achieved by an RMS program  $P$  with around  $|P| = 5120$  multiplications. For this application the BKS scheme requires as parameters  $N = 4096$  and  $\log q = 137$ , which gives a share size of  $3N \log q \approx 210$  kB for each input bit, for a total of 107 MB of communication to each server. In our scheme, choosing  $\gamma = 1.0001$  allows us to use  $N = 2048$ ,  $\log q = 81$ . This results in an input share size of 60.7 kB and a total of 31 MB sent from client to server. Since the (expected) size of the compressed flag sequence is  $|H(\text{pos})| = \gamma \log |\text{pos}|$  and the output modulus of the query should be  $\beta = M$ , the size of the first output share is  $|y_0| = M\gamma \log(4N|P|) + \log \beta \approx 3.2$  kB and the size of the second output share is  $|y_1| = \gamma^M |y_0| \approx 3.5$  kB. Meanwhile, the output share size in BKS is only  $\log \beta \approx 1.2B$  for both servers. Note that only a single output share is sent from each server to the client, so the bulk of communication lies in the input sharing step for both approaches.

A drawback of our solution is that the size of the output shares grows with the size of the database (linearly for  $\mathcal{P}_0$ , and exponentially with base  $\gamma$  for  $\mathcal{P}_1$ ). However, the communication bottleneck is still the size of the input shares and not of the output, as illustrated in the example above. For more general queries, for which this technique relying on additive reconstruction is not applicable, our scheme again provides an improvement over BKS HSS in both computation and communication costs.

**Acknowledgments.** Thomas Attema was supported by the Vraaggestuurd Programma Cyber Security & Resilience, part of the Dutch Top Sector High Tech Systems and Materials program. Pedro Capitão and Lisa Kohl have been supported by the NWO Gravitation project QSC.

## References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015). <https://eprint.iacr.org/2015/046>
2. Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 57–74. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40041-4\\_4](https://doi.org/10.1007/978-3-642-40041-4_4)
3. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03356-8\\_35](https://doi.org/10.1007/978-3-642-03356-8_35)
4. 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](https://doi.org/10.1007/978-3-642-29011-4_42)

5. Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 209–224. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49096-9\\_9](https://doi.org/10.1007/978-3-662-49096-9_9)
6. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 896–912. ACM Press, October 2018
7. 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, Heidelberg (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_16](https://doi.org/10.1007/978-3-030-26954-8_16)
8. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Orrù, M.: Homomorphic secret sharing: optimizations and applications. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 2105–2122. ACM Press, October/November 2017
9. 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](https://doi.org/10.1007/978-3-662-46803-6_12)
10. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 509–539. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53018-4\\_19](https://doi.org/10.1007/978-3-662-53018-4_19)
11. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: improvements and extensions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1292–1303. ACM Press, October 2016
12. Boyle, E., Gilboa, N., Ishai, Y.: Group-Based Secure Computation: Optimizing Rounds, Communication, and Computation. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 163–193. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56614-6\\_6](https://doi.org/10.1007/978-3-319-56614-6_6)
13. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: Karlin, A.R. (ed.) ITCS 2018, vol. 94, pp. 21:1–21:21. LIPIcs, January 2018
14. Boyle, E., Kohl, L., Scholl, P.: Homomorphic secret sharing from lattices without FHE. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 3–33. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17656-3\\_1](https://doi.org/10.1007/978-3-030-17656-3_1)
15. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 1223–1237. ACM Press, October 2018
16. Chillotti, I., Orsini, E., Scholl, P., Smart, N.P., Van Leeuwen, B.: Scooby: improved multi-party homomorphic secret sharing based on FHE. In: Galdi, C., Jarecki, S. (eds.) International Conference on Security and Cryptography for Networks, pp. 540–563. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-14791-3\\_24](https://doi.org/10.1007/978-3-031-14791-3_24)
17. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Citeseer (1997)
18. Cong, K., et al.: Labeled PSI from homomorphic encryption with reduced computation and communication, pp. 1135–1150. ACM Press (2021)
19. Corrigan-Gibbs, H., Boneh, D., Mazières, D.: Riposte: an anonymous messaging system handling millions of users. In: 2015 IEEE Symposium on Security and Privacy, pp. 321–338. IEEE Computer Society Press, May 2015

20. 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, Part II. LNCS, vol. 12697, pp. 842–870. Springer, Heidelberg (2021). [https://doi.org/10.1007/978-3-030-77886-6\\_29](https://doi.org/10.1007/978-3-030-77886-6_29)
21. Dinur, I., Keller, N., Klein, O.: An optimal distributed discrete log protocol with applications to homomorphic secret sharing. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 213–242. Springer, Heidelberg (2018). [https://doi.org/10.1007/978-3-319-96878-0\\_8](https://doi.org/10.1007/978-3-319-96878-0_8)
22. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 93–122. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53015-3\\_4](https://doi.org/10.1007/978-3-662-53015-3_4)
23. Fazio, N., Gennaro, R., Jafarikhah, T., Skeith III, W.E.: Homomorphic secret sharing from Paillier encryption. In: Okamoto, T., Yu, Y., Au, M.H., Li, Y. (eds.) ProvSec 2017. LNCS, vol. 10592, pp. 381–399. Springer, Heidelberg (2017). [https://doi.org/10.1007/978-3-319-68637-0\\_23](https://doi.org/10.1007/978-3-319-68637-0_23)
24. 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](https://doi.org/10.1007/978-3-642-55220-5_35)
25. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. *Des. Codes Crypt.* **75**(3), 565–599 (2015)
26. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13190-5\\_1](https://doi.org/10.1007/978-3-642-13190-5_1)
27. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38348-9\\_3](https://doi.org/10.1007/978-3-642-38348-9_3)
28. 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, Part I. LNCS, vol. 12696, pp. 678–708. Springer, Heidelberg (2021). [https://doi.org/10.1007/978-3-030-77870-5\\_24](https://doi.org/10.1007/978-3-030-77870-5_24)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC, pp. 84–93. ACM Press, May 2005
30. Roy, L., Singh, J.: Large message homomorphic secret sharing from DCR and applications. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Virtual Event, Part III. LNCS, vol. 12827, pp. 687–717. Springer, Heidelberg (2021). [https://doi.org/10.1007/978-3-030-84252-9\\_23](https://doi.org/10.1007/978-3-030-84252-9_23)
31. Shamir, A.: How to share a secret. *Commun. Assoc. Comput. Mach.* **22**(11), 612–613 (1979)
32. Wang, F., Yun, C., Goldwasser, S., Vaikuntanathan, V., Zaharia, M.: Splinter: practical private queries on public data. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2017), pp. 299–313 (2017)





# Discretization Error Reduction for High Precision Torus Fully Homomorphic Encryption

Kang Hoon Lee and Ji Won Yoon(✉)

School of CyberSecurity, Korea University, Seoul, South Korea  
{hoot55,jiwon.yoon}@korea.ac.kr

**Abstract.** In recent history of fully homomorphic encryption, bootstrapping has been actively studied throughout many HE schemes. As bootstrapping is an essential process to transform somewhat homomorphic encryption schemes into fully homomorphic, enhancing its performance is one of the key factors of improving the utility of homomorphic encryption.

In this paper, we propose an extended bootstrapping for TFHE, which we name it by EBS. One of the main drawback of TFHE bootstrapping was that the precision of bootstrapping is mainly decided by the polynomial dimension  $N$ . Thus if one wants to bootstrap with high precision, one must enlarge  $N$ , or take alternative method. Our EBS enables to use small  $N$  for parameter selection, but to bootstrap in higher dimension to keep high precision. Moreover, it can be easily parallelized for faster computation. Also, the EBS can be easily adapted to other known variants of TFHE bootstrappings based on the original bootstrapping algorithm.

We implement our EBS along with the full domain bootstrapping methods known (FDFB, TOTA, Comp), and show how much our EBS can improve the precision for those bootstrapping methods. We provide experimental results and thorough analysis with our EBS, and show that EBS is capable of bootstrapping with high precision even with small  $N$ , thus small key size, and small complexity than selecting large  $N$  by birth.

**Keywords:** Homomorphic encryption · TFHE · Precision

## 1 Introduction

Fully homomorphic encryption (FHE) is a powerful cryptographic scheme that allows to compute on encrypted data with unlimited depth, without leaking any information about it. Nonetheless, performing homomorphic operations on ciphertext accumulates noise or consumes certain amount of levels, and can only evaluate circuits with bounded depth. Thus to support unlimited level of computation,

---

© IACR 2023. This article is the final version submitted by the authors to the IACR and to Springer-Verlag on 17th February 2023. The version published by Springer-Verlag is available at <https://doi.org/00.00000/0000000000>.

© International Association for Cryptologic Research 2023  
A. Boldyreva and V. Kolesnikov (Eds.): PKC 2023, LNCS 13941, pp. 33–62, 2023.  
[https://doi.org/10.1007/978-3-031-31371-4\\_2](https://doi.org/10.1007/978-3-031-31371-4_2)

these schemes come with an operation called *bootstrapping*, which follows from the blueprint of Gentry [18]. Bootstrapping refreshes a possibly noisy, or level-consumed ciphertext into a fresh ciphertext, and allows further computation.

The FHEW/TFHE [11, 16] style bootstrapping methods are still known to be one of the most efficient bootstrapping methods. These algorithms refers to the *blind rotation* algorithm, which refreshes the noisy ciphertext, and evaluates pre-computed lookup table at the same time. From its name, the blind rotate algorithm rotates a polynomial of degree  $N$ , power-of-2, by certain amount blind-folded, over  $2N$ -th cyclotomic ring. The LUT of a function is encoded as the coefficients of the polynomial, and the blind rotation homomorphically selects the value from the encoded LUT by rotating the polynomial. The amount of rotation is decided by the message encrypted inside the ciphertext, with rounding error added due to the scaling-and-rounding of  $(n + 1)$  coefficients in  $\mathbb{T}$  into  $\mathbb{Z}_{2N}$ . Due to this rounding, bootstrapping in FHEW/TFHE style can only preserve at most  $(\log_2 N + 1)$ -bits of precision of the input ciphertext, and the precision is actually much smaller in practice due to the summation of those rounding errors, restricting the high precision usage of these schemes. Thus, it is believed that one should select huge  $N$  to bootstrap with high precision.

To manage real world applications with low precision ciphertexts, the most familiar, but powerful solution is to decompose the message by some base, and encrypt each of the decomposed message in a single ciphertext. The original binary logic of TFHE is a special case of this decomposition, where the base is 2. Clearly, the smaller the base is, the number of bootstrapping increases for performing arithmetic operations on decomposed ciphertexts. If larger base is used to decrease the number of bootstrapping, the bootstrapping precision works as an upper-bound of the size of the base, as the bootstrapping must preserve precision at least  $\log_2(\text{base})$ . This forces to use larger  $N$ , where one can gain 1 bit of precision by doubling  $N$ . Nonetheless, quasi-linear growth in bootstrapping time is accompanied, and the public key size also doubles. Thus, the most efficient usage of TFHE for large precision and corresponding parameter selection is still an open problem.

## 1.1 Our Contributions and Technical Overview

In this paper, we propose a large precision bootstrapping algorithm for TFHE, which we name it by EBS. Compared to the previous literature of TFHE bootstrapping with large precision, our EBS can bootstrap TFHE ciphertext without enlarging the ring dimension  $N$ . Rather, we can keep  $N$  as small as possible as long as the (bootstrapping) error doesn't damage the message in the MSB part. Working with small  $N$  has lots of advantage in TFHE literature since the time complexity of bootstrapping grows quasi-linearly by  $N$ , and the public key size grows linearly. Thus, it is recommended to use small  $N$  for efficiency, and our EBS can solve that problem, while still preserving the precision.

Our EBS inherits the idea to use larger  $N$  to hold larger information of the ciphertext. Nonetheless, rather than increasing  $N$  itself, we use the fact that  $N$  is selected as a power of 2 in TFHE and its variants. We induce a homomorphism to a larger ring from dimension  $N$  to  $2^\nu N$ , where we call  $\nu \in \mathbb{N}$  the extension

factor. The homomorphism is actually a zero padding, and does not affect the security, nor the information of the ciphertext. With the homomorphism, the bits extracted from the ciphertext increases by  $(\log_2 N + 1)$  bits to  $(\log_2 N + \nu + 1)$  bits, and thus we can bootstrap with additional  $\nu$  bits of precision.

For efficiency reasons, we also use the fact that our induced homomorphism is actually a zero padding. Thanks to the zeros, the bootstrapping in dimension  $2^\nu N$  can be converted to  $2^\nu$  times of parallel bootstrappings in dimension  $N$ . The advantage in this is that we can perform the bootstrappings simultaneously, where we can save much time with proper parallelization. Also, the asymptotic complexity decreases compared to when performing the bootstrapping in dimension  $2^\nu N$ .

Also, we provide a proof-of-concept implementation over the TFHE library [12] along with the three variants of the state-of-the-art full-domain functional bootstrapping algorithms. With our implementation, we provide detailed noise analysis, benchmarks with eight sets of parameters with  $N = 1024, 2048$  and  $4096$  that achieves  $\lambda = 80$  or  $128$  bits of security. We also evaluate functions over the torus and provide detailed precision improvements with four sets of parameters. With our EBS, we show that even with  $N = 1024$  and  $2048$ , it is possible to achieve *over* 8 bits of precision, which is known to be possible with at least  $N = 2^{14} = 16386$ . Thus, with our EBS, we can enhance not only the exact computation of TFHE, but also the approximate computation combined with other homomorphic encryption schemes like in [30].

## 1.2 Related Works

High precision bootstrapping is a common problem throughout homomorphic encryption literature. For approximate homomorphic encryption schemes, the high precision bootstrapping is required to evaluate huge depth circuits such as deep neural network training and inference [25, 26], or retrieve statistical information from a dataset. Starting from the dawn of CKKS bootstrapping of 10 to 15 bit of precision [9], many optimizations and better approximations have been studied, and reached 90 to 110 bit of precision [27], or even higher precision of 420 bits which takes 903 seconds [2]. The bootstrapping in CKKS takes much longer than FHEW/TFHE style bootstrappings, but their SIMD (Single Instruction, Multiple Data) structure enables them to bootstrap multiple messages, and usually presented in terms of amortized latency.

Nonetheless, the FHEW/TFHE style bootstrapping still has its own advantages, its significantly low latency, and its capability to evaluate even nonlinear functions with lookup table evaluation. These versatility even brought bridges to approximate homomorphic encryption schemes, to evaluate polynomial functions by approximate schemes, and bring them to TFHE to evaluate nonlinear functions [5, 30]. The enhancements in the usage of FHEW/TFHE itself has also been studied throughout many works, including the extension of binary keys to general keys (ternary, Gaussian, etc.) [20, 32], or improved FHEW bootstrapping with ring automorphisms [28]. When it come to high precision TFHE, most of the works select decomposition of plaintext message [15, 19, 22, 29, 34], with low

precision TFHE bootstrapping, and no algorithm was known to bootstrap a single ciphertext with large precision except using large  $N$ . Recently, Bergerat et al. [3] proposed an algorithm called WoP-PBS (WithOut Padding - Programmable BootStrapping), to extract each bit of the message as a RGSW ciphertext with circuit bootstrapping [10] and evaluate function with vertically packed lookup table. They did not provide any implementations or noise variance in their work, but it is estimated that their WoP-PBS has noise variance bound larger than our EBS by factor of at least  $O(\kappa N)$  when they bootstrap with  $\kappa$ -bits of precision. Nonetheless, their time complexity is linear to  $\kappa$  while our EBS is exponential. This makes their bootstrapping more efficient when  $\kappa$  is sufficiently large. But until certain level, our EBS is more efficient as the circuit bootstrapping is itself quite costly.

## 2 Preliminaries

### 2.1 Notations

We introduce the notations used throughout this paper. The real torus  $\mathbb{T}$  denotes the real set  $\mathbb{R}/\mathbb{Z}$ , which is also interpreted as a half open interval  $[-\frac{1}{2}, \frac{1}{2})$ . Each set  $\mathbb{R}_N[X]$ ,  $\mathbb{Z}_N[X]$ , and  $\mathbb{T}_N[X]$  denote the set  $\mathbb{R}[X]/\langle X^N + 1 \rangle$ ,  $\mathbb{Z}[X]/\langle X^N + 1 \rangle$ , and  $\mathbb{T}[X]/\langle X^N + 1 \rangle$ .

For a set  $\mathcal{S}$ ,  $x \stackrel{\$}{\leftarrow} \mathcal{S}$  implies that  $x$  is sampled from  $\mathcal{S}$  from uniform distribution. Also, for a distribution  $\mathcal{D}$ ,  $x \leftarrow \mathcal{D}$  implies that  $x$  is sampled from a distribution  $\mathcal{D}$ . Next,  $\text{Err}(c)$  represents the error in the ciphertext  $c$ , and  $\text{Var}(\text{Err}(c))$  denotes the variance of error of the ciphertext  $c$ . The parentheses  $\llbracket a, b \rrbracket$  for  $a, b \in \mathbb{Z}$  denotes the set  $\{x \in \mathbb{Z} \mid a \leq x \leq b\}$ . All indices starts with 0 unless mentioned otherwise.

### 2.2 TFHE Ciphertext

The security of TFHE is based on the hardness of the Learning with Errors (LWE) problem [35] and its ring variant, Ring-Learning with Errors (RLWE) [31, 36]. More precisely, the generalization of those problems over the real torus  $\mathbb{T}$ .

**TLWE.** Let  $n \in \mathbb{N}$  be the TLWE dimension, and  $\sigma_{\text{TLWE}}$  be the standard deviation. Then for a discrete message space  $\mathcal{M} \subset \mathbb{T}$ , the TLWE encryption of a message  $m \in \mathcal{M}$  under the key  $\mathbf{s} \in \mathbb{B}^n$  is

$$\text{TLWE}_{\mathbf{s}}(m) = (\mathbf{a}, b) \in \mathbb{T}^{n+1},$$

with  $\mathbf{a} \stackrel{\$}{\leftarrow} \mathbb{T}^n$ , and  $b = \langle \mathbf{a}, \mathbf{s} \rangle + m + e$  where  $e \leftarrow \mathcal{N}(0, \sigma_{\text{TLWE}})$ . Given arbitrarily many TLWE samples with the key  $\mathbf{s}$ , the (torus) LWE problem [11] assures that if it is  $\lambda$ -secure, it requires at least  $2^\lambda$  operations to distinguish TLWE samples from uniform distribution over  $\mathbb{T}^{n+1}$ , or to find  $\mathbf{s}$ . We now denote that the parameter achieves  $\lambda$ -bit of security if it is  $\lambda$ -secure.

The decryption of a TLWE ciphertext first begins with calculating its phase

$$\varphi_{\mathbf{s}}((\mathbf{a}, b)) = b - \langle \mathbf{a}, \mathbf{s} \rangle,$$

and round it to its closest element in  $\mathcal{M}$ .

**TRLWE.** For  $N = 2^\beta, k \in \mathbb{N}$ , and the standard deviation  $\sigma_{\text{TRLWE}}$ , the TRLWE encryption of a message  $m(X)$  in a discrete message space  $\mathcal{M} \subset \mathbb{T}_N[X]$  under the key  $\mathcal{K} \in \mathbb{B}_N[X]^k$  is

$$\text{TRLWE}_{\mathcal{K}}(m(X)) = (A_0(X), \dots, A_{k-1}(X), B(X)) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X],$$

where  $A_i(X) \stackrel{\$}{\leftarrow} \mathbb{T}_N[X]$ ,  $B(X) = \langle (A_0(X), \dots, A_{k-1}(X)), \mathcal{K} \rangle + m(X) + e(X)$ , with each coefficients of  $e(X)$ ,  $e_i \leftarrow \mathcal{N}(0, \sigma_{\text{TRLWE}})$ . The decryption of a TRLWE ciphertext rounds its phase  $\varphi_{\mathcal{K}}((A_0(X), \dots, A_{k-1}(X), B(X)))$  to the closest element in  $\mathcal{M}$ .

**TRGSW.** Given an integer base  $B_G = 2^\gamma \in \mathbb{N}$  and decomposition length  $\ell_G \in \mathbb{N}$ , first define the gadget matrix  $\mathbf{H}$ .

**Definition 1 (Gadget Matrix).** For an integer base  $B_G = 2^\gamma \in \mathbb{N}$  and decomposition length  $\ell_G \in \mathbb{N}$ , we call  $\mathbf{H}$  the gadget matrix given as

$$\mathbf{H} = \left( \begin{array}{c|cc} 1/B_G & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1/B_G^{\ell_G} & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ 0 & \cdots & 1/B_G \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/B_G^{\ell_G} \end{array} \right) \in \mathbb{T}_N[X]^{(k+1)\ell_G \times (k+1)}$$

The TRGSW ciphertext encrypts a integer polynomial  $q(X) \in \mathbb{Z}_N[X]$ . The TRGSW encryption of  $q(X)$  under the key  $\mathcal{K} \in \mathbb{B}_N[X]^k$  is

$$\text{TRGSW}_{\mathcal{K}}(p(X)) = \mathbf{Z} + \mathbf{H} \cdot q(X) \in \mathbb{T}_N[X]^{(k+1)\ell_G \times (k+1)},$$

where  $\mathbf{Z}$  is a vector of  $(k+1)\ell_G$ -TRLWE $_{\mathcal{K}}(0)$ 's. Chillotti et al. [11] defined an external product  $\boxtimes$  between TRGSW $_{\mathcal{K}}(m_a)$  and TRLWE $_{\mathcal{K}}(m_b)$  ciphertext, which gives

$$\text{TRGSW}_{\mathcal{K}}(m_a) \boxtimes \text{TRLWE}_{\mathcal{K}}(m_b) = \text{TRLWE}_{\mathcal{K}}(m_a \cdot m_b),$$

for  $m_a \in \mathbb{Z}_N[X]$ , and  $m_b \in \mathbb{T}_N[X]$ .

### 2.3 Bootstrapping in TFHE

The *bootstrapping* in TFHE is a homomorphic calculation of (discretized) phase of the TLWE ciphertext, and aims to reduce internal noise of the ciphertext. Moreover, it simultaneously evaluates a look-up table (LUT) of a function over the torus, and is also known as the *functional bootstrapping* [4], or *programmable bootstrapping* [13].

To bootstrap a ciphertext, one needs two kinds of public key, namely the bootstrapping key, and the keyswitch key. We will denote each of them as BSK, and KSK. For two secret keys  $\mathbf{s} \in \mathbb{B}^n$  (TLWE key),  $\mathcal{K} \in \mathbb{B}_N^k[X]$  (TRLWE, TRGSW key), the two public keys are defined as follows:

$$\text{BSK} = \{\text{TRGSW}_{\mathcal{K}}(\mathbf{s}_i)\}_{i \in \llbracket 0, n-1 \rrbracket},$$

$$\text{KSK} = \left\{ \text{TLWE}_{\mathbf{s}} \left( \frac{\mathcal{K}_i}{B_{\text{KS}}^j} \cdot k \right) \right\}_{i \in \llbracket 0, N-1 \rrbracket, j \in \llbracket 1, \ell_{\text{KS}} \rrbracket, k \in \llbracket 0, B_{\text{KS}}-1 \rrbracket},$$

where  $B_{\text{KS}}, \ell_{\text{KS}}$  is the decomposition base, and the length of the keyswitch key. Starting from  $\text{TLWE}_{\mathbf{s}}$  ciphertext  $c = (\mathbf{a}, b)$ , bootstrapping consists of four consecutive procedures.

- **ModSwitch:** transforms  $c = (\mathbf{a}, b) \in \mathbb{T}^{n+1}$  into  $\bar{c} = (\bar{\mathbf{a}}, \bar{b}) \in \mathbb{Z}_{2N}^{n+1}$  by computing  $\bar{\mathbf{a}}_i = \lfloor 2N\mathbf{a}_i \rfloor$  for  $i \in \llbracket 0, n-1 \rrbracket$ , and  $\bar{b} = \lfloor 2Nb \rfloor$ . From [11, 22], the variance after ModSwitch comes with

$$\text{Var} \left( \text{Err} \left( \frac{\bar{\mathbf{a}}}{2N}, \frac{\bar{b}}{2N} \right) \right) \leq \text{Var}(\text{Err}(\mathbf{a}, b)) + \frac{n+1}{48N^2},$$

and we denote

$$V_{\text{MS}} = \frac{n+1}{48N^2}.$$

- **BlindRotate:** homomorphically rotates the (possibly noiseless) TRLWE encryption of the test polynomial  $tv \in \mathbb{T}_N[X]$  by  $-\bar{m} = \langle \bar{\mathbf{a}}, \mathbf{s} \rangle - \bar{b} \pmod{2N}$ . This process be viewed as a function evaluator for a function  $f : \mathbb{Z}_{2N} \rightarrow \mathbb{T}$  by setting  $tv_i = f(i)$  for  $i \in \llbracket 0, N-1 \rrbracket$ . The rotation is done by computing the controlled MUX (CMux)  $n$  times:

$$\text{ACC} \leftarrow \text{TRGSW}_{\mathcal{K}}(\mathbf{s}_i) \boxtimes (X^{\bar{\mathbf{a}}_i} - 1) \cdot \text{ACC} + \text{ACC}.$$

Each execution of the CMux multiplies  $X^{\bar{\mathbf{a}}_i \mathbf{s}_i}$  to the accumulator with a certain level of noise growth. Thus, after the BlindRotate, the accumulator is multiplied by  $X^{\langle \bar{\mathbf{a}}, \mathbf{s} \rangle \pmod{2N}}$  blinfolded, and outputs the rotated TRLWE ciphertext  $\text{TRLWE}(X^{-\bar{m}} \cdot tv)$ . After BlindRotate, the variance of ACC is bounded by

$$\begin{aligned} \text{Var}(\text{Err}(\text{ACC})) &\leq \text{Var}(\text{Err}(\text{ACC}_{\text{init}})) \\ &+ n \left( (k+1)N\ell_{\text{BS}} \left( \frac{B_{\text{BS}}}{2} \right)^2 \text{Var}(\text{Err}(\text{BSK})) + \frac{1+kN}{4 \cdot B_{\text{BS}}^{2\ell_{\text{BS}}}} \right), \end{aligned}$$

where the result comes from [11]. Note that  $B_{\text{BS}}, \ell_{\text{BS}}$  is the decomposition base, and the length of the bootstrapping key. Usually, we start off with a noiseless accumulator with  $\text{Var}(\text{Err}(\text{ACC}_{\text{init}})) = 0$ . We now denote

$$V_{\text{BR}} = n \left( (k+1)N\ell_{\text{BS}} \left( \frac{B_{\text{BS}}}{2} \right)^2 \text{Var}(\text{Err}(\text{BSK})) + \frac{1+kN}{4 \cdot B_{\text{BS}}^{2\ell_{\text{BS}}}} \right).$$

- **SampleExtract**: extracts TLWE ciphertext from the rotated TRLWE accumulator ACC. Considering  $tv_i$  as the  $i$ -th coefficient of  $tv$ , it gives  $\text{TLWE}_{\mathcal{K}'}(tv_{\bar{m}}) = \text{TLWE}_{\mathcal{K}'}(f(\bar{m}))$  (resp.  $\text{TLWE}_{\mathcal{K}'}(-tv_{\bar{m}-N}) = \text{TLWE}_{\mathcal{K}'}(f(\bar{m}-N))$ ) if  $\bar{m} \in \llbracket 0, N-1 \rrbracket$  (resp.  $\bar{m} \in \llbracket N, 2N-1 \rrbracket$ ) under the key  $\mathcal{K}' \in \mathbb{B}^{kN}$ . The **SampleExtract** does not accumulate any noise to the ciphertext, so the variance maintains the same.
- **KeySwitch**: converts the key  $\mathcal{K}'$  of extracted  $c' = \text{TLWE}_{\mathcal{K}'}(m)$  into  $\mathbf{s}$ , and gives  $c = \text{TLWE}_{\mathbf{s}}(m)$  that encrypts the same message. Since the **KeySwitch** adds noise to the ciphertext, there have been attempts to remove the **KeySwitch** error by eliminating the need for **KeySwitch** by using  $\mathbf{s} = \mathcal{K}'$  [22], or by moving around the **KeySwitch** before **BlindRotate** [3, 6]. Here, we refer to the TLWE-to-TLWE **KeySwitch**, and the error accumulation is given as

$$\text{Var}(\text{Err}(c)) \leq R^2 \text{Var}(\text{Err}(c')) + kN\ell_{\text{KS}} \text{Var}(\text{Err}(\text{KSK})) + \frac{1}{12} kNB_{\text{KS}}^{-2\ell_{\text{KS}}},$$

where  $R$  is the Lipschitz constant for functional public keyswitching (in our work,  $R = 1$ ). We now denote

$$V_{\text{KS}} = kN\ell_{\text{KS}} \text{Var}(\text{Err}(\text{KSK})) + \frac{1}{12} kNB_{\text{KS}}^{-2\ell_{\text{KS}}}.$$

Algorithm 1 sums up the gate bootstrapping procedure from [11], mainly used to refresh the ciphertext after homomorphic operations (e.g. Homomorphic NAND). From Algorithm 1, the variance of the error of the output ciphertext  $c$  is given by

$$\text{Var}(\text{Err}(c)) \leq V_{\text{BR}} + V_{\text{KS}},$$

and we denote  $V_{\text{BS}} = V_{\text{BR}} + V_{\text{KS}}$ .

## 3 Modified TFHE Bootstrapping

### 3.1 Functional Bootstrapping

Functional bootstrapping, which is the generalization of the gate bootstrapping in Algorithm 1, evaluates the LUT of the target function  $f : \mathbb{T} \rightarrow \mathbb{T}$ , and gives the  $\text{TLWE}_{\mathbf{s}}$  encryption of  $f(\frac{\bar{m}}{2N})$  for  $\bar{m} \in [0, N-1]$ . The procedure is depicted in Algorithm 2. Here, we name it by the *half-domain* functional bootstrapping since it only uses the half of the torus  $[0, \frac{1}{2})$  due to the negacyclic **BlindRotate** (i.e., it gives the encryption of  $-f(\frac{\bar{m}-N}{2N})$  if  $\bar{m} \in [N, 2N-1]$ ). Thus, the domain

---

**Algorithm 1:** Gate Bootstrapping Algorithm (from [11])

---

**Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_s(m \cdot \frac{1}{2})$  with  $m \in \mathbb{B}$ **Input:** Bootstrapping key BSK**Input:** Keyswitch key KSK**Output:** Refreshed TLWE ciphertext  $\text{TLWE}_s((-1)^m \cdot \mu)$ 

- 1  $\bar{\mathbf{a}}_i = \lfloor 2N\mathbf{a}_i \rfloor$  for  $i \in \llbracket 0, n-1 \rrbracket$ , and  $\bar{b} = \lfloor 2Nb \rfloor \quad \triangleright \text{ModSwitch}((a, b), 2N)$
  - 2 Let  $tv = X^{\frac{N}{2}} \cdot (1 + X + \dots + X^{N-1}) \cdot \mu$  for  $\mu \in \mathbb{T}$
  - 3 Let  $\text{ACC}_{init} = (\mathbf{0}, tv) \in \text{TRLWE}_{\mathcal{K}}(tv)$
  - 4  $\text{ACC}_{BR} \leftarrow \text{BlindRotate}((\bar{\mathbf{a}}, \bar{b}), \text{BSK}, \text{ACC}_{init})$
  - 5  $c' \leftarrow \text{SampleExtract}(\text{ACC}_{BR}) \quad \triangleright \text{Extract TLWE}_{\mathcal{K}}((-1)^m \cdot \mu)$
  - 6 **return**  $c = \text{KeySwitch}(c', \text{KSK}) \quad \triangleright \text{TLWE}_s((-1)^m \cdot \mu)$
- 

---

**Algorithm 2:** Half-Domain Functional Bootstrapping (from [4, 19])

---

**Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_s(m)$  with  $m \in [0, \frac{1}{2})$ **Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$ **Input:** Bootstrapping key BSK**Input:** Keyswitch key KSK**Output:** Refreshed TLWE ciphertext  $\text{TLWE}_s(f(\frac{\bar{m}}{2N}))$ 

- 1  $(\bar{\mathbf{a}}, \bar{b}) = \text{ModSwitch}((\mathbf{a}, b), 2N) \quad \triangleright \bar{m} = \bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle \bmod 2N$
  - 2 Let  $tv = \sum_{i=0}^{N-1} f(\frac{i}{2N}) X^i$
  - 3 Let  $\text{ACC}_{init} = (\mathbf{0}, tv) \in \text{TRLWE}_{\mathcal{K}}(tv)$
  - 4  $\text{ACC}_{BR} \leftarrow \text{BlindRotate}((\bar{\mathbf{a}}, \bar{b}), \text{BSK}, \text{ACC}_{init}) \quad \triangleright \text{ACC}_{BR} = \text{TRLWE}(X^{-\bar{m}} \cdot tv)$
  - 5  $c' \leftarrow \text{SampleExtract}(\text{ACC}_{BR}) \quad \triangleright \text{Extract TLWE}_{\mathcal{K}}(f(\frac{\bar{m}}{2N}))$
  - 6 **return**  $c = \text{KeySwitch}(c', \text{KSK}) \quad \triangleright \text{TLWE}_s(f(\frac{\bar{m}}{2N}))$
- 

can be naturally extended to the full torus for any negacyclic function  $h(x) = -h(x + \frac{1}{2})$ .

In Proposition 1, we analyze the error of the functionally bootstrapped ciphertext with  $\mathcal{L}$ -Lipschitz function  $f$  evaluated on an arbitrary message  $m \in [0, \frac{1}{2})$ . From the result, we observe that the rounding error from the `ModSwitch` affects the value of the function itself by directly changing the message of the original ciphertext. The rounding error is thus highly related to the maximal precision a ciphertext can have, and has been pointed out to be the major reason for the severe precision loss in TFHE based applications [13, 22].

Nonetheless, this rounding error was not a serious problem for the *gate* bootstrapping since it only needed 1-bit of precision after the `ModSwitch` to assure its correctness. However, functional bootstrapping works on larger plaintext space



$\mathcal{M}$ , which usually has the size of power of 2 (i.e.,  $|\mathcal{M}| = 2^\pi$ ). Thus, to correctly bootstrap a ciphertext  $\text{TLWE}_s(m)$  with full precision  $\pi$  (with high probability), the errors should satisfy

$$\epsilon_{pre}, \epsilon_{BS} \leq \frac{1}{2|\mathcal{M}|},$$

with high probability for pre-BootStrap and BootStrap errors  $\epsilon_{pre}, \epsilon_{BS}$ .

**Proposition 1 (Functional Bootstrapping Error).** *Let  $c$  be the output of functional bootstrapping with a  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$ . Then the variance of the error between  $\varphi_s(c) = f(m + \epsilon_r) + \epsilon_{BS}$  and  $f(m)$  is bounded by*

$$\text{Var}(\varphi_s(c) - f(m)) \leq \mathcal{L}^2 V_{MS} + V_{BS}.$$

*Proof.* During the ModSwitch, the message  $m$  is rounded into  $\frac{\tilde{m}}{2N} = m + \epsilon_{MS}$ , where  $\epsilon_{MS}$  is the rounding error. Then during the BlindRotate, the function  $f$  is evaluated on  $\frac{\tilde{m}}{2N}$ , with the BlindRotate error  $\epsilon_{BR}$ . Thus, after the KeySwitch, the phase of the output ciphertext  $c$  from line 6 of Algorithm 2 will be

$$\varphi_s(c) = f(m + \epsilon_{MS}) + \epsilon_{BR} + \epsilon_{KS},$$

where  $\epsilon_{KS}$  is the error from the KeySwitch. Then by the  $\mathcal{L}$ -Lipschitz condition, we have

$$\begin{aligned} \text{Var}(\varphi_s(c) - f(m)) &\leq \text{Var}(f(m + \epsilon_{MS}) - f(m)) + \text{Var}(\epsilon_{BR}) + \text{Var}(\epsilon_{KS}) \\ &\leq \text{Var}(\mathcal{L}\epsilon_{MS}) + V_{BR} + V_{KS} \\ &\leq \mathcal{L}^2 \text{Var}(\epsilon_{MS}) + V_{BS} \\ &\leq \mathcal{L}^2 V_{MS} + V_{BS}. \end{aligned}$$

### 3.2 Large Precision TFHE with Functional Bootstrapping

We revise the major branches of TFHE based applications which attempts to operate with large precision. The functional bootstrapping plays an essential role in all of these works, and sometimes appropriately modified to fulfill their required functionality.

**Radix-Based Decomposition with Multiple Ciphertexts.** In this branch, the plaintext  $m$  is decomposed into several digits  $(m_0, m_1, \dots, m_d)$  of certain base(s)  $B$ , and each  $m_i$ 's are encrypted as a single TLWE ciphertext. Usually, small power-of-2 integers ( $2^\pi = 2^1, 2^2$ ) are used as a base [3, 7, 11, 15, 19, 37], or decomposed by co-prime integer bases for the CRT representation [3, 24].

To collaborate with the vector of ciphertexts (i.e., addition, multiplication, function evaluation), *tree-based* [19] and *chaining* [7] method are known as two major solutions. Both methods lookup to huge LUT (encoded in multiple TRLWE ciphertexts) by applying the functional bootstrapping consecutively. The *chaining* method is known to have lower complexity and output noise compared to the

*tree-based* method, but can only evaluate restricted types of function. Recently, Clet et al. [15] generalized the *chaining* method and enabled to evaluate any function by the cost of larger plaintext modulus (i.e.,  $3 \cdot \lceil \log_2 B \rceil + 1$  bits, or  $2 \cdot \lceil \log_2 B \rceil + 1$  bits with additional bootstrapping) to work with base  $B$ .

**Using Larger  $N$ .** As the variance of the rounding error was bounded by  $\text{Var}(\epsilon_{\text{MS}}) \leq \frac{n+1}{48N^2}$ , if we double  $N$ , the bound halves [7, 22, 24, 30]. However, using large  $N$  brings superlinear growth in **BlindRotate** due to the expensive polynomial multiplication with complexity  $O(N \log N)$ . Moreover by doubling  $N$ , the size of the public key (e.g., **BSK**, **KSK**, etc.) exactly doubles which can be a burden for both the client and the server using TFHE based applications.

**Small Hamming Weight  $\text{Ham}(s)$ .** Similar to the case of using large  $N$ , by restricting the hamming weight of  $h = \text{Ham}(s)$ , the rounding error gets bounded by  $\epsilon_r \leq \frac{h+1}{4N}$  [8, 24]. Nonetheless, large TLWE dimension  $n$  is required to achieve the same security level with small hamming weight compared to when using uniform binary key, worsening the performance of bootstrapping as well as its output noise.

**VP-LUT Evaluation and Circuit Bootstrapping.** Recent approach of Bergerat et al. [3] employed the TLWE-to-TRGSW *circuit bootstrapping* from Chillotti et al. [10]. For ciphertext(s) with total  $\kappa$  bit of precision, their method extracts a single bit TLWE encryption with  $\kappa$  functional bootstrappings (i.e.,  $m = \sum_{i=0}^{\kappa-1} m_i \cdot 2^i$  extracted into  $\text{TLWE}_s(\frac{m_i}{2^{\kappa-i}})$ 's). Then the circuit bootstrapping transforms each ciphertexts into  $\text{TRGSW}_{\mathcal{K}}(m_i)$ 's. These TRGSW ciphertexts are used to evaluate the VP-LUT (Vertical Packing LUT), which costs  $\frac{2^\kappa}{N} + \log_2 N - 1$  CMux evaluations. Note that the circuit bootstrapping before the VP-LUT is a costly operation that contains multiple functional bootstrappings, and the output error of the VP-LUT evaluation can be larger than works featured above.

Aforementioned approaches can be combined together for further improvements if needed (e.g., selecting larger  $N$  to attain larger plaintext modulus for the *chaining* method). However, this can reduce the overall usability of TFHE, and should be selected with care.

### 3.3 Extended BlindRotate for Larger Precision

In this section, we introduce a strategy that can be adapted during the **BlindRotate** that allows to attain full precision a single ciphertext can have, even when using small  $N$ . In other words, our method can make the error from the **ModSwitch** quite negligible without using larger  $N$ , and enlarge the precision as long as it is affected the after-bootstrap error,  $\epsilon_{\text{BS}}$ .

The main idea of our algorithm is to *crank up* the **BlindRotate** into a larger auxiliary ring dimension of  $2^\nu N$  using a homomorphism, which is actually sent back to  $2^\nu$ -rings of dimension  $N$  for efficient calculation. We first investigate

on how to *crank up* the **BlindRotate** into a larger space. Note that here, we use an uppercase to clarify the polynomial dimension of TRLWE ciphertext. To be specific,  $\text{TRLWE}_{\mathcal{K}}^N$  implies that each of the polynomial elements in this TRLWE ciphertext is an element of  $\mathbb{T}_N[X]$ , while  $\text{TRLWE}_{\mathcal{K}}^{2^\nu N}$  comes from  $\mathbb{T}_{2^\nu N}[X]$ .

**BlindRotate in Larger Dimension.** Recall that the main precision drop comes from the **ModSwitch**, where the elements of  $\text{TLWE}_s(m) = (\mathbf{a}, b)$  are rounded into  $\mathbb{Z}_{2N}$ . A simple intuition is to *pretend* we are using  $2^\nu N$  instead of  $N$  for  $\nu \in \mathbb{N} \cup \{0\}$ , and round the coefficients into  $\mathbb{Z}_{2^{\nu+1}N}$ . The variance of error from the **ModSwitch** can now be written as

$$\text{Var} \left( \text{Err} \left( \frac{\bar{\mathbf{a}}}{2^{\nu+1}N}, \frac{\bar{b}}{2^{\nu+1}N} \right) \right) \leq \text{Var}(\text{Err}(\mathbf{a}, b)) + \frac{n+1}{48 \cdot 2^{2\nu}N^2},$$

where the variance decreased by  $V_{\text{MS}}/2^{2\nu}$ . What is left is on how to evaluate the **BlindRotate** on dimension  $2^\nu N$ . Thus, we induce a module homomorphism  $\iota : \mathbb{T}_N[X] \rightarrow \mathbb{T}_{2^\nu N}[X]$  by

$$\begin{aligned} \iota : \mathbb{T}_N[X] &\longrightarrow \mathbb{T}_{2^\nu N}[X], \\ p(x) = \sum_{i=0}^{N-1} p_i X^i &\longmapsto p_{\text{ext}}(X) = \sum_{i=0}^{N-1} p_i X^{2^\nu i}, \end{aligned}$$

which is actually a zero padding. We now write the undercase *ext* to denote the polynomials zero-padded in a similar way. By applying  $\iota$  on each torus polynomials of ciphertext, which we denote by  $\iota(\text{TRLWE}_{\mathcal{K}}(p(X)))$  and  $\iota(\text{TRGSW}_{\mathcal{K}}(q(X)))$ , are also extended to

$$\begin{aligned} \iota \left( \text{TRLWE}_{\mathcal{K}}^N(p(X)) \right) &= \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(p_{\text{ext}}(X)) \text{ for } p(X) \in \mathbb{T}_N[X], \\ \iota \left( \text{TRGSW}_{\mathcal{K}}^N(q(X)) \right) &= \text{TRGSW}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(q_{\text{ext}}(X)) \text{ for } q(X) \in \mathbb{Z}_N[X], \end{aligned}$$

for extended key  $\mathcal{K}_{\text{ext}} \in \mathbb{B}_{2^\nu N}[X]$ . Also, since  $\iota$  does not add any noise, the noise variance of the ciphertext stays the same. The external product  $\boxtimes$  follows naturally

$$\begin{aligned} \text{TRGSW}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(q_{\text{ext}}(X)) \boxtimes \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(p(X)) \\ = \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(p(X) \cdot q_{\text{ext}}(X)). \end{aligned}$$

Thanks to the zero padding in TRGSW ciphertext, the error propagation of the external product is exactly the same with computing the external product in dimension  $N$  whether the TRLWE message  $p(X)$  is an extended polynomial or not. Thus, by extending the bootstrapping key BSK with  $\iota$ , we can evaluate the **BlindRotate** with reduced **ModSwitch** error with exactly same error propagation, i.e.,  $V_{\text{BR}}$ . Note that the test vector of the accumulator should be generated in  $\mathbb{T}_{2^\nu N}[X]$ , so the accumulator is a TRLWE encryption under the key  $\mathcal{K}_{\text{ext}}$ , but the message is not an extended torus polynomial.

After the extended `BlindRotate`, the `SampleExtract` follows. However, due to the extension, the `SampleExtract` now gives  $\text{TLWE}_{\mathcal{K}'_{\text{ext}}} = (\mathbf{a}, b) \in \mathbb{T}^{2^\nu kN} \times \mathbb{T}$ . Notice that the key  $\mathcal{K}'_{\text{ext}} \in \mathbb{B}^{2^\nu kN}$  is just a TLWE representation of the extended key  $\mathcal{K}_{\text{ext}}$ , and are all 0 except for the indices multiple of  $2^\nu$ . Thus we extract only the  $2^\nu i$ -th coefficients from  $\mathbf{a}$  for  $i \in \llbracket 0, kN - 1 \rrbracket$  and attain  $\text{TLWE}_{\mathcal{K}'}$ , which can now be keyswitched. The full Algorithm is depicted in Algorithm 3.

---

**Algorithm 3:** Large Precision Bootstrapping (without parallelization)

---

**Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_s(m)$  with  $m \in [0, \frac{1}{2}]$   
**Input:** extension factor  $\nu \in \mathbb{N} \cup \{0\}$   
**Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$   
**Input:** *Extended* Bootstrapping key  $\text{BSK}_{\text{ext}} = \left\{ \text{TRGSW}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(\mathbf{s}_i) \right\}_{i \in \llbracket 0, n-1 \rrbracket}$   
**Input:** Keyswitch key  $\text{KSK}$   
**Output:** Refreshed TLWE ciphertext  $\text{TLWE}_s\left(f\left(\frac{\bar{m}}{2^{\nu+1}N}\right)\right)$

- 1  $(\bar{\mathbf{a}}, \bar{b}) = \text{ModSwitch}((\mathbf{a}, b), 2^{\nu+1}N) \quad \triangleright \bar{m} = \bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle \bmod 2^{\nu+1}N$
- 2 Let  $tv = \sum_{i=0}^{2^\nu N-1} f\left(\frac{i}{2^{\nu+1}N}\right) X^i$
- 3 Let  $\text{ACC} = (\mathbf{0}, tv) \in \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(tv)$
- 4  $\text{ACC}_{\text{BR}} \leftarrow \text{BlindRotate}((\bar{\mathbf{a}}, \bar{b}), \text{BSK}_{\text{ext}}, \text{ACC}) \quad \triangleright \text{ACC}_{\text{BR}} = \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(X^{-\bar{m}} \cdot tv)$
- 5  $c' \leftarrow \text{SampleExtract}(\text{ACC}_{\text{BR}})$
- 6 **return**  $c = \text{KeySwitch}(c', \text{KSK}) \quad \triangleright \text{TLWE}_s\left(f\left(\frac{\bar{m}}{2^{\nu+1}N}\right)\right)$

---

**Parallelization of Extended BlindRotate.** Still, the extended `BlindRotate` contains lots of polynomial multiplications in dimension  $2^\nu N$ , which is quite costly. Therefore, we bring back the calculation of `BlindRotate` multiple polynomial multiplications in dimension  $N$ , which can be easily parallelized. First, we introduce a module isomorphism  $\tau : \mathbb{T}_{2^\nu N}[X] \rightarrow \mathbb{T}_N^{2^\nu}[X]$  defined by

$$\begin{aligned} \tau : \mathbb{T}_{2^\nu N}[X] &\rightarrow \mathbb{T}_N^{2^\nu}[X] \\ p(x) = \sum_{i=0}^{2^\nu N-1} p_i X^i &\mapsto \left( p^{(0)}(X), \dots, p^{(2^\nu-1)}(X) \right) \\ &= \left( \sum_{i=0}^{N-1} p_{2^\nu i} X^i, \dots, \sum_{i=0}^{N-1} p_{2^\nu i + 2^\nu - 1} X^i \right). \end{aligned}$$

Then for  $\text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}$  ciphertext encrypted under extended key  $\mathcal{K}_{\text{ext}}$ , we apply  $\tau$  on each torus polynomial elements in  $\mathbb{T}_{2^\nu N}[X]$ , creating  $(k+1)$  vectors of torus polynomials in  $\mathbb{T}_N^{2^\nu}[X]$ . Due to the zero padding in  $\mathcal{K}_{\text{ext}}$ , collecting  $i$ -th entries from the  $(k+1)$  vectors naturally induces a  $\text{TRLWE}_{\mathcal{K}}^N$  ciphertext for  $i \in \llbracket 0, 2^\nu - 1 \rrbracket$ . We denote the whole process by  $\tau\left(\text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(m)\right)$ :

$$\tau\left(\text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(m)\right) = \left(\text{TRLWE}_{\mathcal{K}}^N(m_0), \dots, \text{TRLWE}_{\mathcal{K}}^N(m_{2^\nu-1})\right).$$

for  $m \in \mathbb{T}_{2^\nu N}[X]$  and  $\tau(m) = (m_0, \dots, m_{2^\nu-1})$ . Since  $\tau$  is rearrangement of coefficients, the noise variance of TRLWE ciphertexts generated by  $\tau$  is at most the noise variance of original ciphertext  $\text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(m)$ .

Then with two constraints that the  $\text{TRGSW}_{\mathcal{K}}^{2^\nu N}(z)$  encrypts an integer  $z \in \mathbb{Z}$  (which is quite common in TFHE literature) and that the  $\text{TRGSW}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(z)$  is extended from  $\text{TRGSW}_{\mathcal{K}}^N(z)$ , we can perform the parallel external product on the decomposed TRLWE ciphertext by

$$\begin{aligned} \text{TRGSW}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(z) \boxtimes \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(m) & \\ & \cong \left( \text{TRGSW}_{\mathcal{K}}^N(z) \boxtimes \text{TRLWE}_{\mathcal{K}}^N(m_0), \dots, \right. \\ & \qquad \qquad \qquad \left. \text{TRGSW}_{\mathcal{K}}^N(z) \boxtimes \text{TRLWE}_{\mathcal{K}}^N(m_{2^\nu-1}) \right) \\ & \cong \left( \text{TRLWE}_{\mathcal{K}}^N(z \cdot m_0), \dots, \text{TRLWE}_{\mathcal{K}}^N(z \cdot m_{2^\nu-1}) \right), \end{aligned}$$

where each external product in dimension  $N$  can be all performed in parallel. Moreover, with the inverse mapping  $\tau^{-1}$ , the output exactly maps to

$$\tau^{-1} \left( \text{TRLWE}_{\mathcal{K}}^N(z \cdot m_0), \dots, \text{TRLWE}_{\mathcal{K}}^N(z \cdot m_{2^\nu-1}) \right) = \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(z \cdot m),$$

for  $z \in \mathbb{Z}$  and  $m \in \mathbb{T}_{2^\nu N}[X]$ . Keeping this in mind, we now suggest a parallelized extended `BlindRotate`, which we now denote as `ExtBlindRotate` in Algorithm 4. From our construction, the  $2^k$  external products in line 6 used to compute the `CMux` gate can be computed in parallel.

*Remark 1.* In Algorithm 4, the rotation of the accumulator was represented by rotating in large dimension, and sending back to its vector of dimension  $N$  with the isomorphism  $\tau$ . We used this representation for simplification, which in practice can actually be *rotated* by changing the order of polynomial vector, and rotating the polynomials.

With the parallel `ExtBlindRotate`, we now present our final extended bootstrapping algorithm `EBS` in Algorithm 5.

**Proposition 2 (EBS).** *The EBS in Algorithm 5 with the extension factor  $\nu \in \mathbb{N} \cup \{0\}$  allows to bootstrap a ciphertext with reduced `ModSwitch` error with variance  $\frac{1}{2^{2\nu}} V_{\text{MS}}$ . The variance of error of the bootstrapped ciphertext is exactly same as  $V_{\text{BS}}$ .*

*Proof.* From line 1 and 2 of Algorithm 5, the reduced `ModSwitch` error variance naturally follows

$$\begin{aligned} \text{Var} \left( \text{Err} \left( \frac{\bar{\mathbf{a}}}{2^{\nu+1}N}, \frac{\bar{b}}{2^{\nu+1}N} \right) \right) & \leq \text{Var}(\text{Err}(\mathbf{a}, b)) + \frac{n+1}{48 \cdot 2^{2\nu}N^2} \\ & \leq \text{Var}(\text{Err}(\mathbf{a}, b)) + \frac{V_{\text{MS}}}{2^{2\nu}}. \end{aligned}$$

**Algorithm 4:** Parallel ExtBlindRotate

---

**Input:**  $(\bar{\mathbf{a}}, \bar{\mathbf{b}}) \in \mathbb{Z}_{2^{\nu+1}N}^n \times \mathbb{Z}_{2^{\nu+1}N}$   
**Input:** extension factor  $\nu \in \mathbb{N} \cup \{0\}$   
**Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$   
**Input:** Bootstrapping key BSK  
**Output:**  $\overrightarrow{\text{ACC}}$  with  $\tau^{-1}(\overrightarrow{\text{ACC}}) = \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(X^{-\bar{\mathbf{b}} + \langle \bar{\mathbf{a}}, \mathbf{s} \rangle \bmod 2^{\nu+1}N} \cdot tv)$

- 1 Let  $tv = \sum_{i=0}^{2^\nu N - 1} f\left(\frac{i}{2^{\nu+1}N}\right) X^i$
- 2  $\overrightarrow{\text{ACC}} \leftarrow \tau\left(\text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^k N}(X^{-\bar{\mathbf{b}}} \cdot tv)\right)$
- 3 **for**  $i \in \llbracket 0, n-1 \rrbracket$  **do**
- 4      $\overrightarrow{\text{RotACC}} \leftarrow \tau\left(X^{\bar{\mathbf{a}}_i} \cdot \tau^{-1}(\overrightarrow{\text{ACC}})\right)$
- 5     **for**  $j \in \llbracket 0, 2^\nu - 1 \rrbracket$  **do**
- 6          $\overrightarrow{\text{ACC}}_j \leftarrow \text{BSK}_i \boxtimes \left(\overrightarrow{\text{RotACC}}_j - \overrightarrow{\text{ACC}}_j\right) + \overrightarrow{\text{ACC}}_j$    ▷ Parallel comp.
- 7     **end**
- 8 **end**
- 9 **return**  $\overrightarrow{\text{ACC}}$

---

Now we show the correctness of ExtBlindRotate in Algorithm 4 and analyze its error propagation. Starting from line 4 of Algorithm 4, we see that it is a rotation of  $\overrightarrow{\text{ACC}}$  by  $\bar{\mathbf{a}}_i$  in  $\mathbb{T}_{2^\nu N}[X]$  and does not add any noise since it only rearranges the coefficients.

In line 6 of Algorithm 4, the CMux gate is evaluated on each row of the accumulator using the external product. Thus if  $\text{BSK}_i$  encrypts 1, the  $\bar{\mathbf{a}}_i$ -rotated accumulator  $\overrightarrow{\text{RotACC}}$  is selected for the next accumulator. If not (i.e., if  $\text{BSK}_i$  encrypts 0),  $\overrightarrow{\text{ACC}}$  is selected. Thus after each  $i$ -th loop, the merged accumulator  $\tau^{-1}(\overrightarrow{\text{ACC}})$  is rotated by  $X^{\bar{\mathbf{a}}_i \mathbf{s}_i}$ , and hence encrypts  $X^{-\bar{\mathbf{b}} + \sum_{p=0}^i \bar{\mathbf{a}}_p \cdot \mathbf{s}_p} \cdot tv$  for  $i \in \llbracket 0, n-1 \rrbracket$ .

For the error propagation of ExtBlindRotate, the errors only comes from the CMux evaluation. More specifically, from the decomposition of the TRLWE ciphertext for the external product, and the external product itself. Thus the error propagation for a single CMux evaluation is exactly the same as the BlindRotate error  $V_{\text{BR}}$ . After the ExtBlindRotate, the error is once more accumulated from the KeySwitch in line 5 of Algorithm 5, whose variance is same as  $V_{\text{KS}}$ . Thus the variance of error after bootstrapping is exactly bounded by  $V_{\text{BS}}$ .

*Remark 2.* The homomorphism  $\iota$  and isomorphism  $\tau$  was defined on module since  $\mathbb{T}_N[X]$ ,  $\mathbb{T}_{2^\nu N}[X]$  are not rings. Nonetheless, this can be easily associated to rings, using the isomorphism between  $\mathbb{Z}_q$  and  $\frac{1}{q}\mathbb{Z}_q \subset \mathbb{T}$ . Thus, our method can naturally be used in ring-based TFHE bootstrapping implementations like in [24, 30, 37].

**Algorithm 5: EBS**


---

**Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_{\mathbf{s}}(m)$  with  $m \in [0, \frac{1}{2})$   
**Input:** extension factor  $\nu \in \mathbb{N} \cup \{0\}$   
**Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$   
**Input:** Bootstrapping key BSK  
**Input:** Keyswitch key KSK  
**Output:** Refreshed TLWE ciphertext  $\text{TLWE}_{\mathbf{s}}(f(\frac{\bar{m}}{2^{\nu+1}N}))$

- 1  $(\bar{\mathbf{a}}, \bar{b}) = \text{ModSwitch}((\mathbf{a}, b), 2^{\nu+1}N) \quad \triangleright \bar{m} = \bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle \bmod 2^{\nu+1}N$
- 2  $\overrightarrow{\text{ACC}} \leftarrow \text{ExtBlindRotate}((\bar{\mathbf{a}}, \bar{b}), \nu, f, \text{BSK})$
- 3  $c' \leftarrow \text{SampleExtract}(\overrightarrow{\text{ACC}}_0) \quad \triangleright \text{Extract TLWE}_{\mathcal{K}'}(f(\frac{\bar{m}}{2^{\nu+1}N}))$
- 4 **return**  $c = \text{KeySwitch}(c', \text{KSK}) \quad \triangleright \text{TLWE}_{\mathbf{s}}(f(\frac{\bar{m}}{2^{\nu+1}N}))$

---

**3.4 Large Precision in Full Domain TFHE Bootstrapping**

Recall that the aforementioned functional bootstrapping algorithms (including our EBS) only works with half domain of the torus  $[0, \frac{1}{2})$  to evaluate arbitrary function  $f : \mathbb{T} \rightarrow \mathbb{T}$ . These algorithms consumes 1 additional bit in front of the MSB of the message, and bootstrapping requires  $p + 1$  bits of precision to successfully bootstrap messages of  $p$  bit precision.

Luckily, it is always possible to evaluate arbitrary function  $f : \mathbb{T} \rightarrow \mathbb{T}$  in the *full* domain of the torus with extra operations. From the state-of-the-art full domain functional bootstrapping algorithms, we observed that our EBS can cooperate with most of these algorithms [14, 15, 24, 37], as they all contain the `ModSwitch` to  $2N$  or  $N$ . The `WoP-PBS` from [3] uses the functional bootstrapping for extracting bits from the ciphertext and during the circuit bootstrapping. Nonetheless, neither the bit extraction nor the the circuit bootstrapping require high precision. As a result, even if our EBS is adaptable, there would be no need to adapt it to their method. Thus, we first briefly explain and compare three full domain bootstrapping algorithms from [15, 24, 37].

**FDfB.** The idea of full domain bootstrapping of FDFB [24] is to select between two test vectors  $p_+, p_- \in \mathbb{T}_N[X]$  based on the sign of message `ct` encrypts. The selection is done by public Mux evaluation, `PubMux`, with the external product. First, the sign of `ct` is first encrypted in a TRGSW-like ciphertext with the circuit bootstrapping [11] (like) procedure. We refer to it as a (T)RLev ciphertext [14], which equals to the last  $\ell_{\text{PM}}$  rows of the TRGSW ciphertext. Note that the multiplication between a torus polynomial  $p(X)$  and a TRLev encryption of  $q(X) \in \mathbb{Z}_N[X]$  outputs a TRLWE encryption of  $q(X) \cdot p(X)$ .

The transformation to TRLev starts with  $\ell_{\text{PM}}$ -functional bootstrappings to extract the sign from `ct`. Then each ciphertexts are TLWE-to-TRLWE keyswitched, which we denote it as `RS`. For specific information about the algorithm, refer to Algorithm 2 of [11]. This ends the conversion to TRLev, and the error in TRLev is bounded by

$$\text{Var}(\text{Err}(\text{TRLev}(\text{sign}(\text{ct})))) \leq V_{\text{BS}} + V_{\text{RS}},$$

and where  $V_{\text{RS}}$  denotes

$$V_{\text{RS}} = n\ell_{\text{RS}}\text{Var}(\text{Err}(\text{RSK})) + \frac{1}{12}nB_{\text{RS}}^{-2\ell_{\text{RS}}}.$$

The TRGSW' encrypts 1 (resp. 0) if the sign of  $\text{ct}$  is positive (resp. negative). Then the evaluation of the PubMux follows by

$$\text{ACC} = (\mathbf{0}, p_+ - p_-) \square' \text{TRLev}(\text{sign}(\text{ct})) + (\mathbf{0}, p_-).$$

Then ACC is initialized as a  $\text{TRLWE}_{\mathcal{K}}$  encryption of  $p_+$  or  $p_-$  according to the sign of the input ciphertext  $\text{ct}$ . The error of ACC is given as

$$\text{Var}(\text{Err}(\text{ACC})) \leq N\ell_{\text{PM}} \left( \frac{B_{\text{PM}}}{2} \right)^2 (V_{\text{BS}} + V_{\text{RS}}) + \frac{1 + kN}{4 \cdot B_{\text{PM}}^{2\ell_{\text{PM}}}},$$

which we will now denote it as  $V_{\text{FDFB-ACC}}$ . What is left is to bootstrap  $\text{ct}$  with the accumulator, and the final error after bootstrap is bounded by

$$V_{\text{FDFB-ACC}} + V_{\text{BS}}.$$

The full algorithm of FDFB(-EBS) is shown in Algorithm 6.

**TOTA.** The main intuition for TOTA [37] bootstrapping is the ModSwitch to  $\mathbb{Z}_N$ , since  $N$  is the maximal number of coefficients a test vector  $tv \in \mathbb{T}_N[X]$  can hold. This makes the quadruple growth to the variance of ModSwitch (i.e.,  $4V_{\text{MS}}$ ). Also, the decryption in  $\mathbb{Z}_{2N}$  during BlindRotate with elements that lied in  $\mathbb{Z}_N$  adds unwanted term  $pN$  to the message, for  $p \in \{0, 1\}$ . Thus TOTA computes the  $pN$  with the sign bootstrapping. The term  $pN$  is removed by subtracting the ModSwitch-ed sign bootstrapped ciphertext, which adds additional ModSwitch error  $V_{\text{MS}}$  and the bootstrapping error  $V_{\text{BS}}$  before the final bootstrapping. After removing the  $pN$ , the ciphertext is then finally bootstrapped with the test vector encoding the function  $f$ .

To sum up, TOTA involves two bootstrappings. First bootstrapping to calculate  $pN$  with pre-bootstrapping error variance

$$\leq V_{\text{ct}} + 4V_{\text{MS}},$$

followed by the second bootstrapping to evaluate the function  $f$ , with pre-bootstrapping error variance

$$\leq V_{\text{BS}} + V_{\text{ct}} + 5V_{\text{MS}}.$$

The variance of error of the output ciphertext is bounded by  $V_{\text{BS}}$ . The full algorithm of TOTA-EBS is shown in Algorithm 7. For the EBS in line 3 of Algorithm 7, the ciphertext  $(\bar{\mathbf{a}}, \bar{\mathbf{b}})$  is already in  $\mathbb{Z}_{2^{\nu+1}N}$ , and we assume that ModSwitch is skipped during the EBS in Algorithm 5.



**Algorithm 6: FDFB-EBS**


---

**Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_s(m)$   
**Input:** extension factor  $\nu \in \mathbb{N} \cup \{0\}$   
**Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$   
**Input:** Bootstrapping key BSK  
**Input:** Keyswitch key KSK  
**Input:** TLWE-to-TRLWE Keyswitch key RSK  
**Input:** PubMux parameter  $\ell_{\text{PM}}, B_{\text{PM}}$   
**Output:** Refreshed TLWE ciphertext  $\text{TLWE}_s\left(f\left(\frac{\widehat{m}}{2^{\nu+1}N}\right)\right)$

- 1 **for**  $i \in \llbracket 1, \ell_{\text{PM}} \rrbracket$  **do**
- 2  $(\mathbf{a}^i, b^i) \leftarrow \text{EBS}\left((\mathbf{a}, b), \nu, \frac{1}{2B_{\text{PM}}^i} f_{\text{sign}}, \text{BSK}, \text{KSK}\right) + \left(\mathbf{0}, \frac{1}{2B_{\text{PM}}^i}\right)$   
 $\triangleright (\mathbf{a}^i, b^i) = \text{TLWE}_s\left(\frac{\text{sign}(\text{ct})}{B_{\text{PM}}^i}\right)$
- 3  $\text{PubACC}_i \leftarrow \text{RS}_{\mathbf{s} \rightarrow \mathcal{K}}((\mathbf{a}^i, b^i), \text{RSK}) \quad \triangleright \text{PubACC} = \text{TRLev}_{\mathcal{K}}(\text{sign}(\text{ct}))$
- 4 **end**
- 5  $\overrightarrow{\text{ACC}} \leftarrow \text{PubMux}(\text{PubACC}, f(x), -f(x - \frac{1}{2}))$   
 $\triangleright \tau^{-1}(\overrightarrow{\text{ACC}}) = \text{TRLWE}_{\mathcal{K}_{\text{ext}}}^{2^\nu N}(tv_{\text{sign}(\text{ct})})$
- 6 **for**  $i \in \llbracket 0, n-1 \rrbracket$  **do**
- 7  $\overrightarrow{\text{RotACC}} \leftarrow \tau\left(X^{\mathbf{a}^i} \cdot \tau^{-1}(\overrightarrow{\text{ACC}})\right)$
- 8 **for**  $j \in \llbracket 0, 2^\nu - 1 \rrbracket$  **do**
- 9  $\overrightarrow{\text{ACC}}_j \leftarrow \text{BSK}_i \boxplus \left(\overrightarrow{\text{RotACC}}_j - \overrightarrow{\text{ACC}}_j\right) + \overrightarrow{\text{ACC}}_j$
- 10 **end**
- 11 **end**
- 12  $c' \leftarrow \text{SampleExtract}(\overrightarrow{\text{ACC}}_0)$
- 13 **return**  $c = \text{KeySwitch}(c', \text{KSK})$

---

**Comp.** Using the fact that every function  $f$  can be written as the sum of (pseudo) odd and even functions, the **Comp** method [15] decomposes a function  $f$  as the sum of odd and even functions  $f_g$ , and  $f_h$ . They aim to compute odd/even functions within 2 bootstrappings each, which can be performed in parallel, and combine them together with simple addition. In total, they can compute any function with 4 functional bootstrappings. The first bootstrapping contains pre-bootstrapping error bounded by

$$\leq V_{\text{ct}} + V_{\text{MS}},$$

and the second bootstrapping bounded by

$$\leq V_{\text{BS}} + V_{\text{MS}}.$$

Due to the addition of two ciphertexts encrypting the value of the odd/even function, the final error of the **Comp** method is bounded by  $2V_{\text{BS}}$ . The full algorithm of **Comp(-EBS)** is shown in Algorithm 8.

**Algorithm 7: TOTA-EBS****Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_s(m)$ **Input:** extension factor  $\nu \in \mathbb{N} \cup \{0\}$ **Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$ **Input:** Bootstrapping key BSK**Input:** Keyswitch key KSK**Output:** Refreshed TLWE ciphertext  $\text{TLWE}_s\left(f\left(\frac{\hat{m}}{2^{\nu}N}\right)\right)$ 

- 1  $(\bar{\mathbf{a}}', \bar{b}') = \text{ModSwitch}((\mathbf{a}, b), 2^{\nu}N) \quad \triangleright \hat{m} = \bar{b}' - \langle \bar{\mathbf{a}}', \mathbf{s} \rangle \bmod 2^{\nu}N$
- 2  $(\bar{\mathbf{a}}, \bar{b}) = \text{ModRaise}_{2^{\nu}N \rightarrow 2^{\nu+1}N}((\bar{\mathbf{a}}', \bar{b}')) \quad \triangleright \bar{m} = \hat{m} + p2^{\nu}N \text{ (in } \mathbb{Z}_{2^{\nu+1}N})$
- 3  $\text{ct}_{\text{sgn}} \leftarrow \text{EBS}((\bar{\mathbf{a}}, \bar{b}), \nu, \frac{1}{4}f_{\text{sign}}, \text{BSK}, \text{KSK}) + (\mathbf{0}, \frac{1}{4}) \quad \triangleright \text{ct}_{\text{sgn}} = \text{TLWE}_s\left(\frac{p}{2}\right)$
- 4  $(\mathbf{a}', b') \leftarrow \text{ModSwitch}(\text{ct}_{\text{sgn}}, 2^{\nu+1}N) \quad \triangleright b' - \langle \mathbf{a}', \mathbf{s} \rangle \bmod 2^{\nu+1}N = p2^{\nu}N$
- 5  $(\mathbf{a}, b) = (\mathbf{a}', b') + (\bar{\mathbf{a}}, \bar{b}) \quad \triangleright \mathbf{b} - \langle \mathbf{a}, \mathbf{s} \rangle \bmod 2^{\nu+1}N = \hat{m}$
- 6 **return**  $c = \text{EBS}((\mathbf{a}, b), \nu, f, \text{BSK}, \text{KSK})$

The whole comparison of three full domain bootstrapping algorithms is shown in Table 1, and the functional bootstrapping is denoted as BS, and the keyswitch as KS. Among the three full domain bootstrapping algorithms, TOTA [37] outperforms other two works in terms of number of operations needed, and also the error after the bootstrapping. However, the variance of error from the ModSwitch nearly quadruples, and quintuples compared to other two. This can be effectively mitigated by our EBS, without changing any of the structure of TOTA. Still, our EBS can also be adapted to other two methods as they all inevitably bootstrap with ModSwitch error added.

### 3.5 Probability of Correct Bootstrapping with EBS

As formerly mentioned, TFHE based applications usually works on plaintext space of  $\mathbb{Z}_p$ , with  $p$  an integer [15, 23, 33]. Using the isomorphism between  $\mathbb{Z}_p$  and  $\frac{1}{p}\mathbb{Z}_p$ , the elements  $m \in \mathbb{Z}_p$  is encoded as  $\frac{m}{p} \in \mathbb{T}$  ( $\frac{m}{2p}$  for half domain). Then, to correctly bootstrap a ciphertext, both pre-bootstrap error and the error after bootstrapping must be smaller than  $\frac{1}{2p}$ . For a ciphertext whose error variance is  $V$  and plaintext space  $\mathbb{Z}_p$ , the probability of correct decryption is estimated by

$$p\left(|\text{Err}(\text{ct})| \leq \frac{1}{2p}\right) = \text{erf}\left(\frac{1}{2p\sqrt{2V}}\right),$$

where **erf** is the error function. Thus, starting from the half-domain EBS with extension factor  $\nu$ , the probability of correct bootstrapping is given as

$$p(\text{HDEBS}) \geq \text{erf}\left(\frac{1}{4p\sqrt{2V_{\text{ct}} + \frac{1}{2^{2\nu-1}}V_{\text{MS}}}}\right) \cdot \text{erf}\left(\frac{1}{4p\sqrt{2V_{\text{BS}}}}\right),$$

**Table 1.** Comparison of 3 full-domain bootstrapping algorithms. Here,  $V_{MS} = \frac{n+1}{48N^2}$ ,  $V_{ct}$  is the variance of error of input ciphertext ct. The BS denotes bootstrapping, RS denotes the TLWE-to-TRLWE KeySwitch.

	<b>FDFB [24]</b>	<b>TOTA [37]</b>	<b>Comp [15]</b>
Pre-bootstrap error	$V_{ct} + V_{MS}$	$V_{ct} + 4V_{MS}$ $V_{ct} + V_{BS} + 5V_{MS}$	$V_{ct} + V_{MS}$ $V_{BS} + V_{MS}$
# of operations	$(\ell_{PM} + 1)$ -BS + $\ell_{PM}$ -RS + 1-PubMux	<b>2-BS</b>	4-BS
Error after Bootstrap	$V_{FDFB-ACC} + V_{BS}$	$V_{BS}$	$2V_{BS}$
Parallel Computing	Partial	<b>✗</b>	<b>✓</b>
Compatible with EBS	<b>✓</b>	<b>✓</b>	<b>✓</b>

as for successful bootstrapping, both the pre-bootstrap error, and the after-bootstrap error must both be smaller than  $\frac{1}{2^p}$ . Thus, for other three bootstrappings, FDFB-EBS, TOTA-EBS, Comp-EBS, we have

$$p(\text{FDFB-EBS}) \geq \text{erf} \left( \frac{1}{2^p \sqrt{2V_{ct} + \frac{1}{2^{2\nu-1}} V_{MS}}} \right) \cdot \text{erf} \left( \frac{1}{2^p \sqrt{2V_{FDFB-ACC} + 2V_{BS}}} \right),$$

$$p(\text{TOTA-EBS}) \geq \text{erf} \left( \frac{1}{2^p \sqrt{2V_{ct} + \frac{1}{2^{2\nu-3}} V_{MS}}} \right) \cdot \text{erf} \left( \frac{1}{2^p \sqrt{2V_{ct} + 2V_{BS} + \frac{5}{2^{2\nu-1}} V_{MS}}} \right) \cdot \text{erf} \left( \frac{1}{2^p \sqrt{2V_{BS}}} \right),$$

$$p(\text{Comp-EBS}) \geq \text{erf} \left( \frac{1}{2^p \sqrt{2V_{ct} + \frac{1}{2^{2\nu-1}} V_{MS}}} \right) \cdot \text{erf} \left( \frac{1}{2^p \sqrt{2V_{BS} + \frac{1}{2^{2\nu-1}} V_{MS}}} \right)^2 \cdot \text{erf} \left( \frac{1}{2^p \sqrt{4V_{BS}}} \right).$$

Then, the probability of failure is calculated by subtracting the success rate from 1, e.g.,  $p_{err}(\text{HDEBS}) \leq 1 - p(\text{HDEBS})$ .

*Remark 3.* The above estimation is for when fixed-point arithmetic is used (in which is IND-CPA<sup>D</sup> secure), using  $\mathbb{Z}_p$  as a plaintext space. Thus, the functions are encoded in a staircase-like manner, i.e.,  $\sum_i f(\lfloor \frac{p}{N} \cdot i \rfloor)$  for  $f: \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ , that works like a breakwater to prevent pre-bootstrap noise flooding out.

## 4 Experimental Results

We implemented our (HD)EBS along with the adaptation of EBS to three full-domain bootstrappings, FDFB-EBS, TOTA-EBS, Comp-EBS. Our implementations were built upon TFHE library [12], where the torus elements  $\mathbb{T}$  are represented as 32-bit integer,  $\mathbb{Z}_{2^{32}}$ . Our experiments were executed with Intel i9-13900K running at 5.80 GHz with 24 cores (8 performance cores, 16 efficient

---

**Algorithm 8: Comp-EBS (Modular)**

---

**Input:** TLWE ciphertext  $(\mathbf{a}, b) \in \text{TLWE}_s(m)$ **Input:** extension factor  $\nu \in \mathbb{N} \cup \{0\}$ **Input:** A  $\mathcal{L}$ -Lipschitz morphism  $f : \mathbb{T} \rightarrow \mathbb{T}$ **Input:** Bootstrapping key BSK**Input:** Keyswitch key KSK**Input:** Plaintext modulus  $\mathcal{P}$ **Output:** Refreshed TLWE ciphertext  $\text{TLWE}_s(f(\frac{\hat{m}}{2^\nu N}))$ 

- 1  $c_1 = \text{EBS}((\mathbf{a}, b), \nu, \frac{1}{2^{\mathcal{P}}} + \frac{1}{\mathcal{P}} \lfloor 2^\nu N \mathcal{P} x \rfloor, \text{BSK}, \text{KSK}) - (\mathbf{0}, \frac{1}{2^{\mathcal{P}}})$
  - 2  $c_2 = \text{EBS}((\mathbf{a}, b), \nu, \frac{1}{2^{\mathcal{P}}} + \frac{1}{4} + \frac{1}{\mathcal{P}} \lfloor 2^\nu N \mathcal{P} x \rfloor, \text{BSK}, \text{KSK}) - (\mathbf{0}, \frac{1}{2^{\mathcal{P}}} + \frac{1}{4})$
  - 3  $c_3 = \text{EBS}(c_1, \nu, \frac{f(x) - f(-x - \frac{1}{\mathcal{P}})}{2}, \text{BSK}, \text{KSK})$
  - 4  $c_4 = \text{EBS}(c_2, \nu, \frac{f(x) + f(-x - \frac{1}{\mathcal{P}})}{2}, \text{BSK}, \text{KSK})$
  - 5 **return**  $c_3 + c_4$
- 

cores) and 32 threads, 128 GB RAM, and with 64-bit Ubuntu 22.04 environment. We compiled our experiment with g++ 11.3.0 with flags `-ltfhe-spqlios-fma -fopenmp -lquadmath`, using `spqlios` FFT in TFHE for fast polynomial multiplication, and multi-threading for our parallel EBS. The code we used for experiment is publicly available at <https://github.com/Stirling75/Extended-BootStrapping>.

#### 4.1 TFHE Parameters

As the security of TFHE scheme has its roots in the hardness of (R)LWE problem, its security level is decided by the dimension of ciphertext (i.e.,  $n, kN$ ), and its corresponding standard deviation of errors added during encryption (i.e.,  $\sigma_{\text{TLWE}}, \sigma_{\text{TRLWE}}$ ). The security of TRGSW is guaranteed by the security of TRLWE, as it is a vector of TRLWE ciphertexts. We estimated the cost of attack models for various instances  $(n, \sigma_{\text{TLWE}})$ , and  $(N, \sigma_{\text{TRLWE}})$  with the lattice estimator [1]. For most of our instances, the cost of the dual-hybrid attack [17] were estimated to be the cheapest.

In Table 2, we present eight TFHE parameter sets satisfying  $\lambda = 80$ , 128 bits of security, which implies it requires at least  $2^\lambda$  operations for the attack models to succeed their attacks. As can be seen from parameter set  $\text{I}_1$  and  $\text{III}_1$ , T(R)LWE ciphertexts with same dimension, decreasing the security parameter  $\lambda$  enables to use small standard deviation for the error which decreases the error after bootstrapping. Notice that for parameter sets  $(\text{I}_1, \text{I}_2, \text{I}_3)$  and  $(\text{III}_1, \text{III}_2, \text{III}_3)$ , we used exactly the same parameters except for the ring dimension  $N$ , to observe the effect of using larger  $N$ . Nonetheless, the native TFHE library only supports FFT of dimension 1024, and we made slight changes in their library to enable FFT on dimension 2048 and 4096 to support fast polynomial multiplication.

**Table 2.** TFHE parameter sets.  $\lambda$  indicates the security level of given parameter set.

Param Set	$\lambda$	TLWE		TRLWE			KSK		BSK	
		$n$	$\sigma_{\text{TLWE}} (\log_2)$	$N$	$k$	$\sigma_{\text{TRLWE}} (\log_2)$	$\ell_{\text{KS}}$	$B_{\text{KS}}$	$\ell_{\text{BS}}$	$B_{\text{BS}}$
I <sub>1</sub>	80	750	-21.2	1024	1	-29.3	3	2 <sup>8</sup>	7	2 <sup>4</sup>
I <sub>2</sub>	80	750	-21.2	2048	1	-32	3	2 <sup>8</sup>	7	2 <sup>4</sup>
I <sub>3</sub>	80	750	-21.2	4096	1	-32	3	2 <sup>8</sup>	7	2 <sup>4</sup>
II	80	900	-25.7	2048	1	-32	5	2 <sup>6</sup>	7	2 <sup>4</sup>
III <sub>1</sub>	128	670	-12.4	1024	1	-20.1	3	2 <sup>5</sup>	8	2 <sup>3</sup>
III <sub>2</sub>	128	670	-12.4	2048	1	-32	3	2 <sup>5</sup>	8	2 <sup>3</sup>
III <sub>3</sub>	128	670	-12.4	4096	1	-32	3	2 <sup>5</sup>	8	2 <sup>3</sup>
IV	128	1300	-26.1	2048	1	-32	5	2 <sup>6</sup>	7	2 <sup>4</sup>

However, this modified FFT still uses 64-bit `double` with 53 bits of precision, accumulating non-negligible noise during polynomial multiplication when  $N \geq 2048$ . We found this inhibits exact noise analysis for cases where polynomial multiplication over dimension  $N \geq 2048$  is used. Further details on noise accumulation during FFT and polynomial multiplication can be found in Proposition 1 of [21].

We also describe FDFB parameters in Table 3. These parameters are only used for FDFB and has no effect on other bootstrapping methods. Using these parameters, FDFB runs with  $\ell_{\text{PM}} + 1 = 6$  (functional) bootstrappings,  $\ell_{\text{PM}} = 5$  RS and 1 PubMux operations.

**Table 3.** Parameters for RSK and PubMux for FDFB.

Parameter Set	RSK		PubMux	
	$\ell_{\text{RS}}$	$B_{\text{RS}}$	$\ell_{\text{PM}}$	$B_{\text{PM}}$
I <sub>1</sub> , I <sub>2</sub> , I <sub>3</sub>	6	2 <sup>5</sup>	5	2 <sup>5</sup>
II	4	2 <sup>6</sup>		
III <sub>1</sub> , III <sub>2</sub> , III <sub>3</sub>	4	2 <sup>6</sup>		
IV	6	2 <sup>5</sup>		

## 4.2 Performance Results

With the parameters we suggested in Sect. 4.1, we make a thorough analysis in terms of public key size, latency, and noise.

**Public Key Size.** Following the footsteps of [11], we measure the size of the public keys (BSK, KSK, RSK) published for homomorphic operations. First we measure the size of TLWE, TRLWE, TRGSW ciphertexts and then calculate the size of each public keys. For example, the size of TLWE ciphertext of parameter set  $I_1$  is  $(n + 1) \times 32 = 24\,032$  bits  $\approx 3.004$  KB. Also, since KSK is composed of  $N \times \ell_{KS} \times B_{KS}$  TLWE ciphertexts, the size of the KSK is 2.36 GB.

Likewise, we evaluate the key size for every parameter set and present the result in Fig. 1. As we can observe from the result of parameter sets ( $I_1, I_2, I_3$ ) and ( $III_1, III_2, III_3$ ), the key size doubles as  $N$  doubles. Still, with proper adjustment of parameters, we can make the public key size ‘sufficiently small’ (see BSK and KSK of parameter  $I_1$  and II) even with large  $N$ .

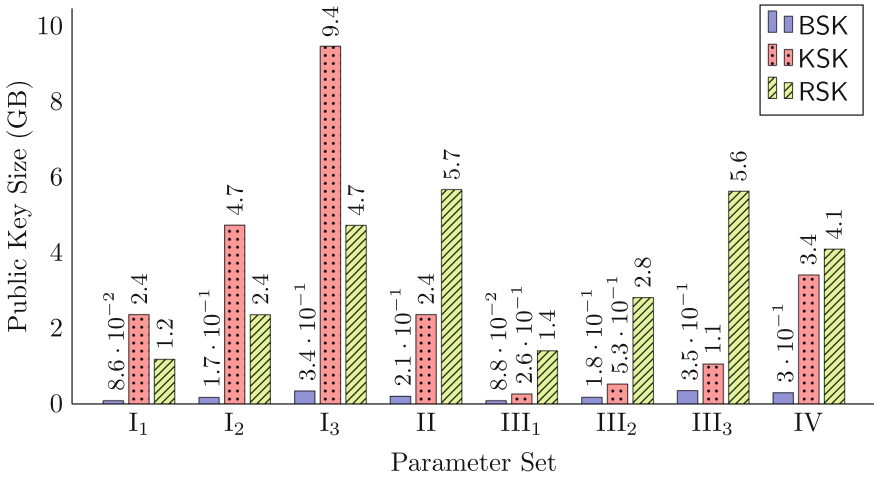


Fig. 1. Public key (BSK, KSK, RSK) sizes for our parameters.

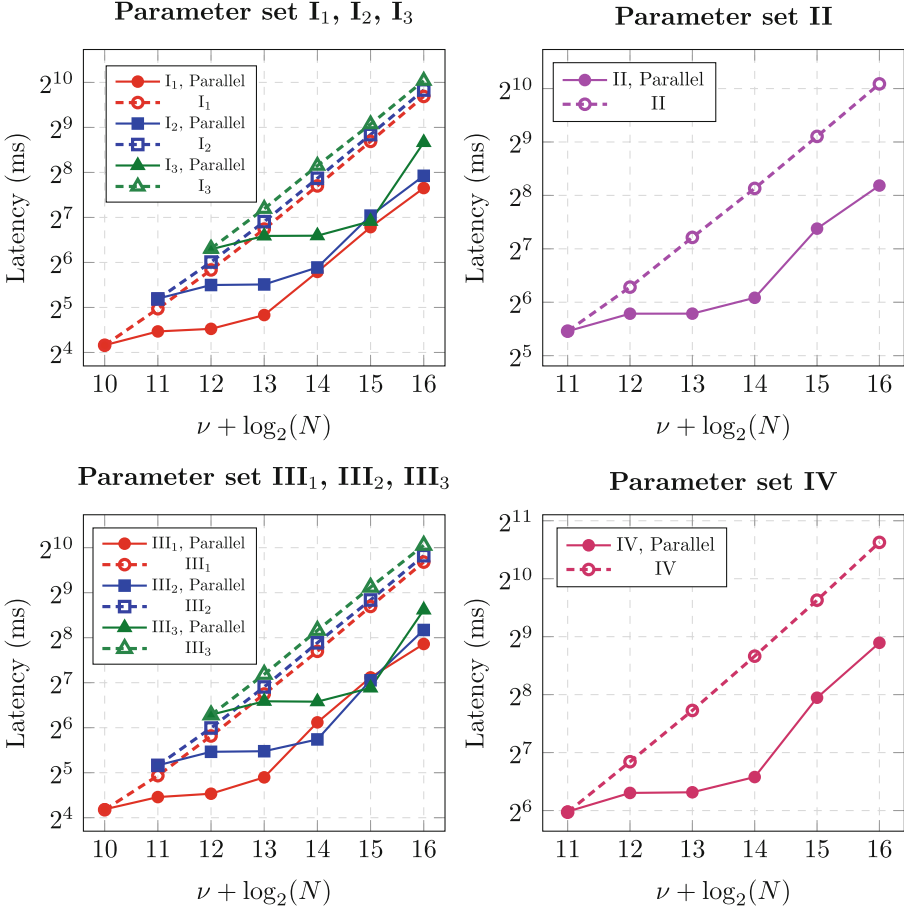
**Noise Analysis.** Next, we examine the variance of noise for our parameters. As our work heavily relies on the noise estimates of variety of algorithms, we found it necessary to show experimental validation of our estimations proposed in previous sections. We present our results in Table 4. We calculated (with label  $(c)$ ) the standard deviations with our variance estimations, and experimentally validated it (with label  $(E)$ ) by observing  $2^{15}$  samples for each case. For bootstrappings, we set the test vector as identity function and calculated the standard deviations.

For ModSwitch, we calculated two standard deviations with the conventional  $V_{MS}$  formula  $\frac{n+1}{48N^2}$ , and with the hamming weight of the TLWE key  $\mathbf{s}$ ,  $\frac{\text{Ham}(\mathbf{s})+1}{48N^2}$ . Our result shows that the experimental error nearly corresponds to the standard deviation calculated with the hamming weight for all cases. For bootstrapping (including TOTA and Comp), we observe that for parameter sets  $I_2$ ,  $I_3$ , II and IV, their experimental bootstrapping error are larger than expected due to the non-negligible FFT error accumulated during polynomial multiplication. Nonetheless, for parameter set  $III_2$  and  $III_3$ , their main error standard deviations are dominated by the keyswitching error and seems to follow the estimation well. We provide detailed error analysis of BlindRotate and KeySwitch in Appendix A.1, Table 5.

To only observe the output noise of each bootstrapping method, we first pre-computed the rotated amount (during BlindRotate) for given ciphertext. Then we bootstrapped the ciphertext with each bootstrapping methods, and then subtracted the rotated test vector from the accumulator, thereby eliminating the effect of the ModSwitch. From the result, FDFB shows larger noise standard deviation compared to other bootstrapping methods due to the PubMux operation. Moreover, from the result of Proposition 1, we claim that until  $\mathcal{L}\sigma_{MS}$ , where  $\mathcal{L}$  is the Lipschitz constant of the function  $f$ , is larger than the output bootstrapping noise standard deviation, there will be room for improvement with our EBS (Fig. 2).

**Table 4.** Estimated noise standard deviation (with label  $^{(c)}$ ) and experimental noise standard deviation (with label  $^{(E)}$ ) of ModSwitch and four bootstrapping methods. The standard deviations are presented in the form of  $\log_2$ .

		$I_1$	$I_2$	$I_3$	II	$III_1$	$III_2$	$III_3$	IV
ModSwitch	$\sigma_{MS}^{(c)}$	-8.016	-9.016	-10.016	-8.885	-8.097	-9.097	-10.097	-8.620
	Ham(s)	379	379	379	448	330	330	330	645
	$\sigma_{MS}^{(c,Ham)}$	-8.508	-9.508	-10.508	-9.387	-8.607	-9.607	-10.607	-9.125
	$\sigma_{MS}^{(E)}$	<b>-8.509</b>	<b>-9.506</b>	<b>-10.507</b>	<b>-9.385</b>	<b>-8.601</b>	<b>-9.591</b>	<b>-10.544</b>	<b>-9.116</b>
Bootstrap	$\sigma_{BS}^{(c)}$	-14.411	-14.855	-14.355	-16.614	-6.000	-6.107	-5.607	-16.364
	$\sigma_{BS}^{(E,id)}$	<b>-14.882</b>	<b>-14.663</b>	<b>-14.033</b>	<b>-15.293</b>	<b>-6.369</b>	<b>-6.157</b>	<b>-5.654</b>	<b>-15.128</b>
FDFB	$\sigma_{FDFB}^{(c)}$	-4.250	-4.194	-3.193	-5.951	4.161	4.554	5.554	-5.703
	$\sigma_{FDFB}^{(E,id)}$	<b>-10.196</b>	<b>-10.010</b>	<b>-8.524</b>	<b>-9.882</b>	<b>-2.152</b>	<b>-1.830</b>	<b>-1.750</b>	<b>-9.753</b>
TOTA	$\sigma_{TOTA}^{(c)}$	-14.411	-14.855	-14.355	-16.614	-6.000	-6.107	-5.607	-16.364
	$\sigma_{TOTA}^{(E,id)}$	<b>-14.879</b>	<b>-14.639</b>	<b>-14.063</b>	<b>-15.628</b>	<b>-6.371</b>	<b>-6.160</b>	<b>-5.649</b>	<b>-14.992</b>
Comp	$\sigma_{Comp}^{(c)}$	-13.911	-14.355	-13.855	-16.114	-5.500	-5.607	-5.107	-15.864
	$\sigma_{Comp}^{(E,id)}$	<b>-14.581</b>	<b>-14.460</b>	<b>-13.611</b>	<b>-14.796</b>	<b>-6.163</b>	<b>-6.149</b>	<b>-5.645</b>	<b>-14.533</b>

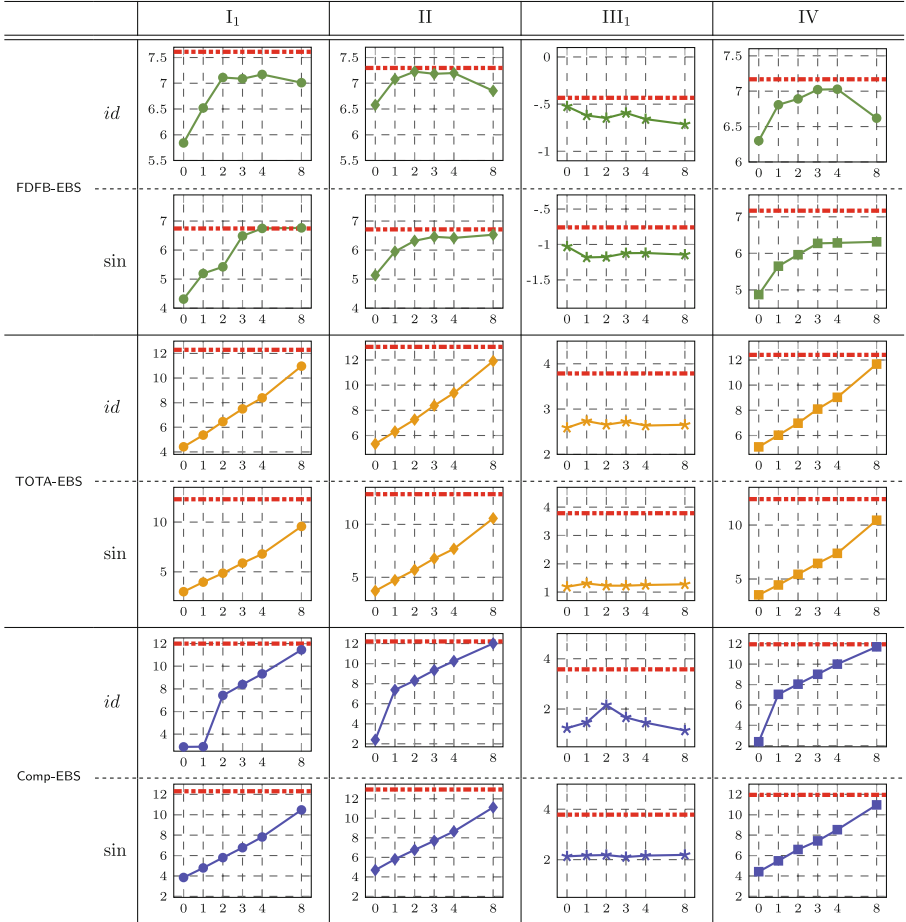


**Fig. 2.** Latency for EBS with eight sets of parameters given in Table 2. The x-axis represents  $\nu + \log_2(N)$  for each parameter set, and the y-axis represents the latency in milliseconds on logarithmic scale of base 2.

**Benchmarks.** We now present benchmarks for EBS, and precision growth when adapted to full-domain bootstrapping methods. We first measure the latency for computing a single conventional TFHE bootstrapping with EBS. Note that when the extension factor  $\nu = 0$ , the EBS becomes exactly same as original bootstrapping. From the results, we can see that for non-parallelized settings (depicted in dashed line), using  $N' = 2^\nu N$  with  $\nu > 0$  is slower than using EBS with dimension  $N$  and extension factor  $\nu$ . Also, since the EBS increases the number of external products by the factor of  $2^\nu$ , it is easy to see the exponential increase in latency with respect to the extension factor  $\nu$ . For parallelized version of EBS (depicted in solid line), we found that we lose full parallelization from  $\nu = 3$  due to computational limitations. Thus, our results show exponential growth after  $\nu = 3$ .



**Precision.** We finally turn to our major contribution of EBS, the precision enhancement. For three full-domain bootstrapping methods we introduced, we measured the output noise standard deviations for two functions, the identity function (with Lipschitz constant  $\mathcal{L}_1 = 1$ ) and  $f(x) = 43 \sin(x/32)$  (with Lipschitz constant  $\mathcal{L}_2 \approx 33.772$ ) by matching the torus  $[-\frac{1}{2}, \frac{1}{2}]$  to  $[-64, 64]$ . With the three sigma rule, we measured the bit precision of the results and presented them in Fig. 3 for four parameter sets I<sub>1</sub>, II, III<sub>1</sub>, IV. With the experimental results from Table 4 and some additional experiments, we also calculated the (ideal)



**Fig. 3.** Experimental output precision of three full-domain bootstrapping methods (FDFB, TOTA, Comp) with EBS evaluated with the four parameter sets I<sub>1</sub>, II, III<sub>1</sub>, and IV. The *id* and *sin* stands for the identity function and the function  $f(x) = 43 \sin(x/32)$ , by mapping the torus to  $[-64, 64]$ . The  $x$ -axis represents the extension factor  $\nu$ , and the  $y$ -axis represents the output bit precision. The red dash-dot line represents the maximum precision for each parameter and bootstrapping method.

maximum precision of each parameter set and depicted it as a dash-dot line in all of the figures. We noticed that the change of function (which changes the test vector) introduces noticeable changes to the maximum precision to FDFB due to their PubMux, but is quite negligible to other two full-domain bootstrapping methods.

From our results, we can see for both parameter sets I<sub>1</sub>, II and IV, the precision improvement is significantly clear for TOTA and Comp. Nonetheless, due to the PubMux in FDFB, the maximum precision for their method is lower than other two methods. For parameter set III<sub>1</sub>, due to its large noise standard deviation ( $\sigma_{\text{TRLWE}} = 2^{-20.1}$ ) for TRGSW ciphertext (since they achieve 128 bits of security), their output precision is quite lower than other parameter sets. In this case, it is suggested to increase the size of  $N$  and use smaller standard deviation, like in parameter set IV ( $N = 2048, \sigma_{\text{TRLWE}} = 2^{-32}$ ).

## 5 Conclusion

In this paper, we suggested a high precision TFHE bootstrapping algorithm EBS, which can almost remove the affect of ModSwitch during bootstrapping. The biggest advantage in our scheme is that it allows to bootstrap with large precision with small public key size compared to when enlarging  $N$ . Also, EBS can be naturally parallelized for fast computation, where no known algorithm is known to parallelize TFHE bootstrapping. Thus, we can even bootstrap much faster than previous literature of using large  $N$ . We show that our EBS is compatible with both modular, and approximate arithmetic, as well as previously known full domain bootstrapping algorithms. We also believe EBS can be one of the solution for bridging other homomorphic encryption schemes with TFHE, by allowing high precision, nonlinear function bootstrapping with small cost.

**Acknowledgements.** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2022R1F1A1074291).

## A Appendix

### A.1 Noise Analysis

We present detailed analysis of noise for the BlindRotate, KeySwitch, and FDFB-ACC in Table 5. Due to the error added during polynomial multiplication (with FFT), the experimental error standard deviation for  $N \geq 2048$  is larger than estimated results.

**Table 5.** Estimated noise standard deviation (with label  $^{(c)}$ ) and experimental noise standard deviation (with label  $^{(E)}$ ). The standard deviations are presented in the form of  $\log_2$ .

		I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	II	III <sub>1</sub>	III <sub>2</sub>	III <sub>3</sub>	IV
Bootstrap	$\sigma_{BR}^{(c)}$	-14.620	-16.771	-16.271	-16.640	-6.406	-14.794	-14.295	-16.374
	$\sigma_{BR}^{(E,id)}$	<b>-15.355</b>	<b>-15.539</b>	<b>-14.675</b>	<b>-15.297</b>	<b>-7.181</b>	<b>-11.539</b>	<b>-10.682</b>	<b>-15.129</b>
	$\sigma_{KS}^{(c)}$	-15.407	-14.907	-14.407	-19.039	-6.607	-6.107	-5.607	-19.439
	$\sigma_{KS}^{(E)}$	<b>-15.413</b>	<b>-14.910</b>	<b>-14.419</b>	<b>-19.068</b>	<b>-6.657</b>	<b>-6.157</b>	<b>-5.655</b>	<b>-19.472</b>
	$\sigma_{BS}^{(c)}$	-14.411	-14.855	-14.355	-16.614	-6.000	-6.107	-5.607	-16.364
	$\sigma_{BS}^{(E,id)}$	<b>-14.882</b>	<b>-14.663</b>	<b>-14.033</b>	<b>-15.293</b>	<b>-6.369</b>	<b>-6.157</b>	<b>-5.654</b>	<b>-15.128</b>
DFDB	$\sigma_{ACC}^{(c)}$	-4.250	-4.194	-3.194	-5.951	4.161	4.554	5.554	-5.703
	$\sigma_{ACC}^{(E,id)}$	<b>-10.207</b>	<b>-10.004</b>	<b>-8.546</b>	<b>-9.892</b>	<b>-2.155</b>	<b>-1.838</b>	<b>-1.751</b>	<b>-9.764</b>
	$\sigma_{DFDB}^{(c)}$	-4.250	-4.194	-3.193	-5.951	4.161	4.554	5.554	-5.703
	$\sigma_{DFDB}^{(E,id)}$	<b>-10.196</b>	<b>-10.010</b>	<b>-8.524</b>	<b>-9.882</b>	<b>-2.152</b>	<b>-1.830</b>	<b>-1.750</b>	<b>-9.753</b>

## A.2 Benchmarks

In this section, we present the benchmark results for our parallelized and non-parallelized EBS along with the benchmarks for three full-domain bootstrapping methods. Note that none of the operations except the EBS were parallelized for fair comparison (Table 6).

**Table 6.** Benchmark results for EBS and three full-domain bootstrapping methods. The **NP** is for non-parallelized, and **P** denotes parallelized results. All results are presented in milliseconds (ms).

		$\nu$	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	II	III <sub>1</sub>	III <sub>2</sub>	III <sub>3</sub>	IV
HDEBS	<b>NP</b>	0	17.88	36.44	79.28	43.98	18.08	35.84	77.88	62.72
		1	31.38	64.4	146.24	77.88	30.58	63.7	145.5	114.74
		2	57.04	120.02	284	148.42	56.32	119.88	286.42	211.9
		3	107	233.82	537.98	280.32	107.52	236.54	556.74	405.54
		4	207.98	458.46	1050.92	550.52	207.48	457.56	1061.12	792.28
		5	412.86	910.26	—	1088.24	414.1	904.76	—	1580.92
	<b>P</b>	0	17.895	36.67	78.4	43.94	18.205	35.85	78.1	63.21
		1	22.12	45.205	96.28	55.16	21.98	44.15	96.12	79.015
		2	23.015	45.6	96.62	55.2	23.15	44.55	95.53	79.64
		3	28.465	59.235	120.6	67.8	29.79	53.42	118.055	95.545
		4	55.305	131.64	406.12	166.36	69.47	132.765	392.385	246.675
		5	110.225	243.07	—	290.92	138.865	288.21	—	475.665
	6	201.275	—	—	—	232.34	—	—	—	

(continued)

**Table 6.** (*continued*)

	$\nu$	$I_1$	$I_2$	$I_3$	II	III <sub>1</sub>	III <sub>2</sub>	III <sub>3</sub>	IV	
DFDB-EBS	<b>P</b>	0	130.22	253.78	543.57	303.67	123.02	243.8	529.18	457.89
		1	154.42	298.65	605.84	355.07	146.28	295.92	600.24	520.72
		2	157.94	307.18	639.79	365.77	150.76	294.27	626.65	532.28
		3	239.23	432.24	1016.78	478.13	193.35	443.41	957.79	821.15
		4	410.85	854.58	2561.31	1012.06	438.65	833.93	2522.98	1484.04
TOTA-EBS	<b>P</b>	0	37.98	75.1	162.64	92.84	37.15	74.9	166.33	134.13
		1	44.06	88.56	191.68	113.51	45.33	91.87	193.88	157.56
		2	46.77	91.86	196.31	113.15	46.52	91.47	196.84	159.61
		3	83.1	134.62	275.54	155.91	59.89	131.12	307.76	258.35
		4	153.99	273.6	827	330.39	139.47	268.45	822.79	474.7
Comp-EBS	<b>P</b>	0	74.67	150.25	325.02	184.9	74.12	149.96	332.31	267.24
		1	89.31	181.11	384.38	226.16	89.9	181.79	394.74	317.37
		2	93.24	184.26	391.46	225.2	93.5	182.38	392.45	318.85
		3	171.22	308.43	561.69	321.24	152.34	271.53	595.7	460.4
		4	303.01	544.15	1650.39	657.29	281.76	541.95	1669.71	964.52

## References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Paper 2015/046 (2015). <https://eprint.iacr.org/2015/046>, <https://eprint.iacr.org/2015/046>
2. Bae, Y., Cheon, J.H., Cho, W., Kim, J., Kim, T.: Meta-BTS: bootstrapping precision beyond the limit. Cryptology ePrint Archive (2022)
3. Bergerat, L., et al.: Parameter optimization & larger precision for (T) FHE. Cryptology ePrint Archive (2022)
4. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Simulating homomorphic evaluation of deep learning predictions. In: Dolev, S., Hendler, D., Lodha, S., Yung, M. (eds.) CSCML 2019. LNCS, vol. 11527, pp. 212–230. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-20951-3\\_20](https://doi.org/10.1007/978-3-030-20951-3_20)
5. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: CHIMERA: combining ring-LWE-based fully homomorphic encryption schemes. J. Math. Cryptol. **14**(1), 316–338 (2020)
6. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 483–512. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-96878-0\\_17](https://doi.org/10.1007/978-3-319-96878-0_17)
7. Bourse, F., Sanders, O., Traoré, J.: Improved secure integer comparison via homomorphic encryption. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 391–416. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-40186-3\\_17](https://doi.org/10.1007/978-3-030-40186-3_17)
8. Carpov, S., Izabachène, M., Mollimard, V.: New techniques for multi-value input homomorphic evaluation and applications. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 106–126. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-12612-4\\_6](https://doi.org/10.1007/978-3-030-12612-4_6)

9. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 360–384. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78381-9\\_14](https://doi.org/10.1007/978-3-319-78381-9_14)
10. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 377–408. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70694-8\\_14](https://doi.org/10.1007/978-3-319-70694-8_14)
11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.* **33**(1), 34–91 (2020)
12. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption library (2016). <https://tfhe.github.io/tfhe/>
13. Chillotti, I., Joye, M., Paillier, P.: Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In: Dolev, S., Margalit, O., Pinkas, B., Schwarzmann, A. (eds.) CSCML 2021. LNCS, vol. 12716, pp. 1–19. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-78086-9\\_1](https://doi.org/10.1007/978-3-030-78086-9_1)
14. Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13092, pp. 670–699. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92078-4\\_23](https://doi.org/10.1007/978-3-030-92078-4_23)
15. Clet, P.E., Zuber, M., Boudguiga, A., Sirdey, R., Gouy-Pailler, C.: Putting up the swiss army knife of homomorphic calculations by means of TFHE functional bootstrapping (2022). <https://eprint.iacr.org/2022/149>
16. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_24](https://doi.org/10.1007/978-3-662-46800-5_24)
17. Espitau, T., Joux, A., Kharchenko, N.: On a dual/hybrid approach to small secret LWE. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) INDOCRYPT 2020. LNCS, vol. 12578, pp. 440–462. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-65277-7\\_20](https://doi.org/10.1007/978-3-030-65277-7_20)
18. Gentry, C.: A fully homomorphic encryption scheme. Stanford university (2009)
19. Guimaraes, A., Borin, E., Aranha, D.F.: Revisiting the functional bootstrap in TFHE. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**(2), 229–253 (2021)
20. Joye, M., Paillier, P.: Blind rotation in fully homomorphic encryption with extended keys. In: Dolev, S., Katz, J., Meisels, A. (eds.) CSCML 2022. LNCS, vol. 13301, pp. 1–18. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-07689-3\\_1](https://doi.org/10.1007/978-3-031-07689-3_1)
21. Klemsa, J.: Fast and error-free negacyclic integer convolution using extended Fourier transform. In: Dolev, S., Margalit, O., Pinkas, B., Schwarzmann, A. (eds.) CSCML 2021. LNCS, vol. 12716, pp. 282–300. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-78086-9\\_22](https://doi.org/10.1007/978-3-030-78086-9_22)
22. Klemsa, J.: Setting up efficient TFHE parameters for multivalued plaintexts and multiple additions. *Cryptology ePrint Archive* (2021)
23. Klemsa, J., Önen, M.: Parallel operations over TFHE-encrypted multi-digit integers. In: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, pp. 288–299 (2022)
24. Kluczniak, K., Schild, L.: FDFB: full domain functional bootstrapping towards practical fully homomorphic encryption. arXiv preprint [arXiv:2109.02731](https://arxiv.org/abs/2109.02731) (2021)

25. Lee, E., et al.: Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In: International Conference on Machine Learning, pp. 12403–12422. PMLR (2022)
26. Lee, J.W., et al.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access* **10**, 30039–30054 (2022)
27. Lee, Y., Lee, J.W., Kim, Y.S., Kim, Y., No, J.S., Kang, H.: High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022. LNCS, vol. 13275, pp. 551–580. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-06944-4\\_19](https://doi.org/10.1007/978-3-031-06944-4_19)
28. Lee, Y., et al.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. *Cryptology ePrint Archive* (2022)
29. Liu, Z., Micciancio, D., Polyakov, Y.: Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. *Cryptology ePrint Archive* (2021)
30. Lu, W.J., Huang, Z., Hong, C., Ma, Y., Qu, H.: PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: 2021 IEEE Symposium on Security and Privacy (SP), pp. 1057–1073. IEEE (2021)
31. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. *J. ACM (JACM)* **60**(6), 1–35 (2013)
32. Micciancio, D., Polyakov, Y.: Bootstrapping in FHEW-like cryptosystems. In: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 17–28 (2021)
33. Okada, H., Kiyomoto, S., Cid, C.: Integer-wise functional bootstrapping on TFHE: applications in secure integer arithmetics. *Information* **12**(8), 297 (2021)
34. Paul, J., Tan, B.H.M., Veeravalli, B., Aung, K.M.M.: Non-interactive decision trees and applications with multi-bit TFHE. *Algorithms* **15**(9), 333 (2022)
35. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM (JACM)* **56**(6), 1–40 (2009)
36. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10366-7\\_36](https://doi.org/10.1007/978-3-642-10366-7_36)
37. Yang, Z., Xie, X., Shen, H., Chen, S., Zhou, J.: TOTA: fully homomorphic encryption with smaller parameters and stronger security. *Cryptology ePrint Archive* (2021)



# Verifiable Capacity-Bound Functions: A New Primitive from Kolmogorov Complexity

(Revisiting Space-Based Security in the Adaptive Setting)

Giuseppe Ateniese<sup>1</sup>, Long Chen<sup>2</sup>, Danilo Francati<sup>3</sup>(✉),  
Dimitrios Papadopoulos<sup>4</sup>, and Qiang Tang<sup>5</sup>

<sup>1</sup> George Mason University, Virginia, USA

ateniese@gmu.edu

<sup>2</sup> Institute of Software, Chinese Academy of Sciences, Beijing, China

chenlong@iscas.ac.cn

<sup>3</sup> Aarhus University, Aarhus, Denmark

dfrancati@cs.au.dk

<sup>4</sup> Hong Kong University of Science and Technology, Kowloon, Hong Kong

dipapado@cse.ust.hk

<sup>5</sup> The University of Sydney, Sydney, Australia

qiang.tang@sydney.edu.au

**Abstract.** We initiate the study of *verifiable capacity-bound function* (VCBF). The main VCBF property imposes a strict lower bound on the number of bits read from memory during evaluation (referred to as minimum capacity). No adversary, even with unbounded computational resources, should produce an output without spending this minimum memory capacity. Moreover, a VCBF allows for an efficient public verification process: Given a proof of correctness, checking the validity of the output takes significantly fewer memory resources, sublinear in the target minimum capacity. Finally, it achieves soundness, i.e., no computation-bounded adversary can produce a proof that passes verification for a false output. With these properties, we believe a VCBF can be viewed as a “space” analog of a verifiable delay function. We then propose the first VCBF construction relying on evaluating a degree- $d$  polynomial  $f$  from  $\mathbb{F}_p[x]$  at a random point. We leverage ideas from *Kolmogorov complexity* to prove that sampling  $f$  from a large set (i.e., for high-enough  $d$ ) ensures that evaluation must entail reading a number of bits proportional to the size of its coefficients. Moreover, our construction benefits from existing verifiable polynomial evaluation schemes to realize our efficient verification requirements. In practice, for a field of order  $O(2^\lambda)$  our VCBF achieves  $O((d + 1)\lambda)$  minimum capacity, whereas verification requires just  $O(\lambda)$ . The minimum capacity of our VCBF construction holds against adversaries that perform a constant number of random memory accesses during evaluation. This poses the natural question of whether a VCBF with high minimum capacity guarantees exists when dealing with adversaries that perform non-constant (e.g., polynomial) number of random accesses.

---

The authors are listed alphabetically.

© International Association for Cryptologic Research 2023

A. Boldyreva and V. Kolesnikov (Eds.): PKC 2023, LNCS 13941, pp. 63–93, 2023.

[https://doi.org/10.1007/978-3-031-31371-4\\_3](https://doi.org/10.1007/978-3-031-31371-4_3)

**Keywords:** Kolmogorov complexity · Adaptive security · Polynomial evaluation · Verifiable computation · Verifiable delay function

## 1 Introduction

Time and space complexity are functions that measure the efficiency of algorithms. These two functions are related (sometimes appear in the same setting) but distinct. For instance, “time” may refer to the number of memory accesses performed by an algorithm, while “space” refers to the amount of memory needed. In general, we try to minimize these functions, i.e., an ideal algorithm is one that is fast and tight. However, in cryptography, we are also interested in algorithms that are deliberately slow or capacious with the idea that, if the adversary must run them, the attack will be slow and costly. This has found numerous applications, e.g., in the context of proof-of-work for distributed consensus [46], and anti-spam mechanisms [9, 27]; and password hashing or key derivations to be used against offline brute-force [38, 52].

The most prominent definitions for “space-demanding” functions proposed in the literature are memory-hardness [2–6, 19, 21, 49], and bandwidth-hardness [15, 54]. While they share the same initial motivation, these notions vary in their formalization and achieved security guarantees. Memory-hardness, as originally defined [49], guarantees a lower bound in the memory/time product required to compute the function. Informally, a function is memory-hard if the product of the evaluation memory cost  $m$  and time  $t$  for any adversary cannot be less than  $mt \in \Omega(n^2)$ , where  $O(n)$  is the time for an honest party. This has been widely proposed as a countermeasure against attackers that aim to gain an unfair advantage by using customized hardware, such as an ASIC, as it forces one to dedicate a significant area of memory to avoid being too slow. Thus, the cost of ASIC manufacturing would grow proportionally. Bandwidth-hardness guarantees that the *energy cost* for evaluating the function does not differ much across different platforms with variable computing energy costs (e.g., CPU vs. ASIC). In practice, this is based on the observation that although ASICs may have superior energy consumption for specific tasks, off-chip memory accesses incur comparable energy costs on ASICs and CPUs. Thus, energy consumption is enforced by ensuring a substantial amount of off-chip memory accesses.

None of these provides a strict bound on the amount of distinct bits read: The former allows for a trade-off between memory block accesses and computing, whereas the latter bounds the ratio of energy consumption benefits for ASIC adversaries. A different notion, predating memory and bandwidth-hardness, is that of memory-bound functions [1, 26, 28] that do impose an expected lower bound on the number of memory accesses, expressed as cache misses.

All these notions have “symmetric” hardness in the following sense. Given a candidate input-output pair  $(x, y)$  for function  $f$ , verifying whether  $f(x) = y$  is, at best, achieved by evaluating  $f$ . In that sense, evaluation and checking require the same amount of resources. In many applications, it would be desirable to have an *efficient public verification* algorithm that can check the correctness of



an evaluation using significantly fewer resources, after the party that evaluates  $f$  provides a proof of correctness  $\pi$  for  $y$ . In practice, considering a cryptographic puzzle application [27, 36, 43], a challenger receiving multiple candidate puzzle solutions from different parties should be able to verify their correctness with much less effort than it took to compute them. Even considering egalitarian proofs of work [12], checking the validity of a proposed evaluation with considerably smaller memory requirements allows for easy validation by numerous lightweight clients.

In the context of time-demanding functions, verifiable delay functions (VDFs) introduced by Boneh et al. [18] achieve such a property: any observer can verify that the computation of the function was performed correctly and can do so efficiently. The scope of this paper is to introduce an analogous function but for capacious/space-hungry algorithms. However, “space” or memory functions appear to be more intricate. Indeed, space-hardness does cover the memory needed by an algorithm for instructions, data, and inputs. Still, as discussed above, hardness often involves a trade-off between space and time, i.e., an algorithm is allowed to use more time to make up for a smaller memory footprint.

**This Work: Verifiable Capacity-Bound Functions.** In this work, we initiate the study of *verifiable capacity-bound functions (VCBF)*. At a high level, a VCBF guarantees: (a) a strict lower bound  $m$  in the necessary number of *distinct bits read* from memory in order to evaluate the function each time (referred to as *minimum capacity complexity*), (b) a public verification process that given a proof  $\pi$  can check the correctness of an evaluation by reading only  $o(m)$  bits, and (c) soundness, i.e., no computationally-bounded adversary should be able to produce a convincing proof for an incorrect evaluation. The space notion of VCBF differs significantly from other space-related functions: It provides a strict lower bound on the number of distinct bits read at each evaluation of the function (minimum capacity) even if an adversary adaptively chooses its strategy after the function is instantiated. In addition, it does not present any time/space trade-off on evaluation, i.e., the only way to compute the VCBF’s output is to satisfy its minimum capacity complexity unless the VCBF is heavily precomputed. Note that every function inevitably presents a time/space trade-off under heavy pre-computation, e.g., evaluate the function on all inputs and store the outputs into an ordered dictionary. This differs from other space functions [1, 4, 6, 28, 54] in which an evaluator can tune the memory usage at the price of computing the function in more time, even if the function has not been preprocessed.

Also, unlike the notion of asymmetric hardness [13] which allows parties with access to a secret trapdoor to evaluate  $f$  quickly, we aim for public verifiability. Hence, a VCBF is a publicly verifiable function that does not present a time/space trade-off on evaluation. In that sense, it can be viewed as a space-analog of a VDF.

### Comparison Between VCBF and Other “Space-Demanding” Functions.

We provide a more detailed discussion of the relation between VCBFs and other primitives that attempt to bound the resources used when evaluating a function.

**Table 1.** Comparison summary between VCBF and existing space-demanding functions. We exclude from the comparison any primitive that deviates from our objectives: (i) primitives based on heuristics or enforce memory/space usage on expectation, i.e., no strict lower bound on the memory/space usage (e.g., [1] and puzzle-based constructions [26,28]) or, (ii) primitives that require interaction (e.g., [7,29,53]). Publicly verifiable means that the correctness of the function’s output can be publicly verified with significantly fewer space-units than evaluating the function. We use the term “memory” to denote the total space required to evaluate the function (this does not guarantee a lower bound on the number of bits read).

	Space-unit per execution	Security analysis	Publicly verifiable
Code-hard functions [13]	Memory	Ideal cipher	✗
Memory-hard functions [4,6,49]	Time/Memory trade-off	ROM & Pebbling	✗
Bandwidth-hard functions [54]	Time/Cache-miss trade-off	ROM & Pebbling	✗
VCBF (this work)	Bits read	Standard	✓

See Table 1 for a comparison summary between VCBF and the most prominent functions and the corresponding space flavors.

*Minimum Number of Computation Steps.* Such primitives provide a lower bound on the minimum number of *sequential steps* necessary. Notable examples include classic time-locked puzzles [55], key-derivation function PBKDF2 [38], and the recently proposed verifiable delay functions mentioned above [18,50,59]. Another related notion is proof-of-sequential-work (PoSW) [23,25,42], which is similar to VDF except PoSW is not a function. Typically, these enforce a repeated operation (hashing or squaring in the group with an unknown order). As discussed, our VCBF shares the same spirit as VDF but for space/energy consumption.

*Minimum Number of Memory Access.* As explained above, memory-bound functions provide an expectation of the lower bound on the number of cache misses for any polynomial-time bounded adversary. In [1,26] a subset of a large random table (thus incompressible) is accessed during evaluation. However, they do not meet our requirement of the strict capacity lower bound on the number of bits read for each evaluation (like VDF for the time setting) since their lower bound is only a statistical expectation.

Follow-up work [28] suggests a construction with a time/space trade-off for the process of constructing the table from a representation, but this permits us to easily trade memory accesses for computation workload. We stress that [26,28] leverage a puzzle-based approach: They reach the desired number of cache misses by evaluating the function multiple times. Hence, they cannot be considered functions due to their puzzle-based nature (similarly to the analogy between VDF and PoW in the time setting). Lastly, [1] leverages an inner function  $f$  whose inverse  $f^{-1}$  cannot be evaluated in less time than accessing the memory. Hence, their construction presents a time/space trade-off: A malicious adversary may choose to involve more time to reduce the number of memory accesses.

*Code-Hard Functions.* Code-hard algorithms [13] require that a minimum amount of memory is used in order to store the code (generated using block ciphers). This has found different applications, e.g., white box encryption [11, 16, 17, 34] or big-key encryption [10]. The key difference between a code-hard function and VCBF is that while a large amount of memory space must be dedicated for storing the code-hard function, it is possible that only a small fraction of those stored bits must be retrieved during evaluation (i.e., using memory does not imply reading bits). A VCBF imposes a non-trivial strict lower bound on bits read from memory during each evaluation.

*Memory and Bandwidth-Hard Functions.* These functions adaptively read/write from/in the memory to achieve two different objectives: Memory-hard functions require evaluators to use a large amount of memory while bandwidth-hard functions produce a high number of cache misses.<sup>1</sup> These functions [4, 6, 49, 54] allow adversaries to dynamically trade additional computation for reduced memory usage on evaluation (even without precomputation); thus, they do not meet our strict lower bound guarantee.<sup>2</sup> Moreover, the existing formalizations are highly reliant on the random oracle model, e.g., [54] for bandwidth hardness and [4, 6, 49] for memory hardness (in the parallel random oracle model). This comes naturally, as many of these works use variations of a graph-pebbling game to model their computation, heuristically estimating the energy cost for each unit computation and memory access operations. On the other hand, our VCBF definition does not rely on the random oracle model (this does not preclude the possibility of specific VCBFs operating in this model). Another impact of relying on the random oracle model is that it makes it harder to design an efficient verification algorithm as it “destroys” any algebraic structures between inputs and outputs.

We stress that a VCBF’s lower bound in memory bits accessed can be used to infer a lower bound in energy consumption, analogous to the motivation behind bandwidth-hard functions. E.g., considering an ASIC-based adversary with on-chip memory of size  $s$  bits (such as a hardware cache) a VCBF that guarantees to access  $m$  bits from main memory imposes a  $u(m - s)$  lower energy consumption, where  $u$  is the atomic cost for reading one bit from memory.

In a recent work [31], the first memory-hard VDF construction was proposed by combining a SNARK with a parallelizable prover with a memory-hard “sequential” function. Although this result is close in spirit with what a VCBF tries to achieve, we do not aim for an explicit time lower bound, whereas

---

<sup>1</sup> We stress that, in the setting of memory-hard functions, the term “memory” is used to denote the number of memory blocks required to correctly evaluate (in a given time) the function. This differs from the VCBF objective of forcing the evaluator to read a fixed number of distinct bits (requiring  $n$  memory blocks of size  $w$  on evaluation does not imply reading  $nw$  distinct bits since multiple memory blocks may present a redundant pattern that may be compressed).

<sup>2</sup> We stress that memory-hard functions present a time/space trade-off on evaluation that varies according to the notion of memory hardness considered (e.g., time-space complexity [49], cumulative space complexity [6], sustained space complexity [4]).

the memory-bound we achieve is strict without leaving room for time/space trade-offs, as explained above.

*Proof of Space (PoSpace).* PoSpace [7, 29, 53] extends memory-hard functions with efficient verification and adopts the graph pebbling framework and the random oracle model. The prover convinces a verifier that it consumed its space capacity to store data while allowing for efficient verification in both space and time. Like memory-hard functions, the PoSpace constructions can only guarantee a time/space trade-off, thus cannot enforce a space lower bound. Also, the security analysis is based on the heuristic (parallel) random oracle model.

**Overview of Techniques.** The main challenge in building a VCBF is finding a function that has a natural strict lower bound on the space necessary for evaluation while still allowing for efficient verification. Past works [2, 3, 5, 6, 15, 19, 21, 54] achieve the first property only on expectation (i.e., expected lower bound) by relying on assumptions such as the random oracle or ideal cipher. Hence, this approach fails to achieve a strict lower bound and makes it harder to achieve the second property as it dismantles structured relations between the function’s inputs and outputs that could be used for efficient verification.

In this work, we deviate from previous techniques significantly. To model the inability of an adaptive space-based adversary to compute an output without reading enough data from memory, we turn our attention to *Kolmogorov complexity* [40], which measures the complexity (in an absolute sense) of an object in terms of the minimum number of bits necessary to represent it. Kolmogorov complexity is viewed as a fundamental theory of computer science and has been shown connected with multiple areas in cryptography [41, 45, 56]. (The most recent work of Liu and Pass [41] proves the equivalence of a computational bounded version of the Kolmogorov complexity and the existence of one-way functions.) Somewhat more formally, the Kolmogorov complexity of object  $x$  is the minimum number of bits needed to represent *any description*  $(T, \alpha)$  where  $T$  is a Turing machine and  $\alpha$  is a string such that  $T(\alpha)$  outputs  $x$ . One can view  $T$  as an adaptive decompressing algorithm and  $\alpha$  as a “compression” computed adaptively from  $x$ . Based on this, our first observation is that if an algorithm *depends* on an object  $x$  (e.g.,  $x$  could be the description of the algorithm itself or the algorithm’s input), then its execution *cannot require reading fewer bits than the Kolmogorov complexity of  $x$* . In that sense, Kolmogorov complexity is the right tool for us; choosing a function with high Kolmogorov complexity readily provides an arguably loose bound for the minimum capacity of a VCBF even in the presence of an adaptive adversary that chooses its strategy (that determines how memory is read and organized) after the function is instantiated (see Sect. 1.1 for a discussion about adaptive security in the space setting).

On the other hand, when building our VCBF we need to identify a function that is amenable to verification; ideally, it should preserve an efficiently checkable (algebraic) relation between inputs and outputs. One candidate function is *polynomial evaluation* for single-variable polynomial  $f(X) \in \mathbb{F}_p[x]$  of degree  $d$  of the form  $f(X) = \sum_{i=0}^d a_i \cdot x^i$ . The good news is that there exist numerous

works in the literature for verifiable polynomial evaluation (e.g., [30, 32, 48, 61]). In order to use such a scheme for a VCBF we need to ensure it is *publicly verifiable* (anyone can verify it using public parameters) and *publicly delegatable* (anyone can query it on an evaluation point). In our construction, we use the lightweight scheme of Elkhiyaoui et al. [30]. Its verification process requires a constant number of operations among a constant number of elliptic curve elements. This is important for us since we want VCBF to have verification capacity complexity *sublinear* in its evaluation’s minimum capacity. Using [30], the latter is  $O((d+1)\lambda)$  whereas the former is  $O(\lambda)$  (where  $\lambda$  is the security parameter), i.e., the gap is linear in the degree of the polynomial.

The “honest” way of evaluating polynomial  $f(X)$  is by reading its coefficients  $a_i$ , so by fixing  $|(a_0, \dots, a_d)| \geq m$  (where  $|x|$  denotes the bit length of  $x$ ) one would hope to get a VCBF with minimum capacity  $m$ . However, this is not the case as every polynomial has multiple alternative representations that an adversary may try to exploit in order to bypass the memory capacity bound. For example, all lists of the form  $(x_0, \dots, x_d), (f(x_0), \dots, f(x_d))$ , for any choice of  $d+1$  distinct  $x_i$ , completely determine the coefficients  $(a_0, \dots, a_d)$  of  $f(X)$  (by interpolating the points). Here is where Kolmogorov complexity comes in handy: The above evaluations and points together with a Turing machine that performs polynomial interpolation are a valid description, in terms of Kolmogorov complexity, of the coefficients  $(a_0, \dots, a_d)$ . As a consequence, it *cannot be significantly shorter* than the Kolmogorov complexity  $C(a_0, \dots, a_d)$  of the coefficients of the polynomial  $f(X)$  (Theorem 5).

What remains is to find a way to sample a polynomial  $f(X)$  with high Kolmogorov complexity. For any large-enough set, most of its elements have sufficiently high Kolmogorov complexity. Since this holds for arbitrary sets, sampling at random from a large-enough set of polynomials guarantees that the chosen polynomial is of high Kolmogorov complexity with high-enough probability.

As discussed above, many previous works inherently adopt non-standard models in their definitions to capture the fact that a function is memory-heavy (e.g., random-oracle, ideal cipher, or heuristic assumptions about graph pebbling). Instead, we want to base our security definition in the standard setting, and we regard our paper on VCBF as a foundational one. Our approach is to model adversaries as Turing machines that read (at most) a fixed number of distinct bits  $m$  (whose value is estimated using the Kolmogorov complexity) from a precomputed memory  $\tau$  of size  $n \geq m$  (Sect. 4). We stress that it is crucial to consider the memory of size  $n$  larger than  $m$  since an adversary can leverage a large memory to increase its advantage  $\epsilon$  while, at the same time, minimizing the number  $m$  of distinct bits it must read to answer a particular challenge (for example, it can store a large dictionary containing several evaluations of the polynomial  $f(X)$ ). However, this introduces the new challenge of estimating the adversary’s advantage  $\epsilon$  with respect to the memory size  $n$ : A particularly challenging task when working in the standard model with black-box access to the adversary. In more detail, it is hard to provide a strict bound on the number of (partial) information that can be stored in a memory of size  $n$  since their space

requirement highly depends on the precomputation strategy (e.g., the entropy of the precomputed values) and the encoding (e.g., memory organization, memory access patterns) that can be adaptively chosen by an adversary after some parameters are revealed (e.g., the object to compress). Still, we show that it is possible to give a positive, meaningful estimation of  $\epsilon$  and  $n$  when considering adversaries that perform a constant number  $v \in O(1)$  of random accesses (e.g., conditional jumps) in order to read discontinuous bits from memory. We discuss the formulation of our definition and our results in Sect. 4 and Sect. 5.1, respectively.

**Summary of our Contributions.** Our contributions in this work can be summarized as follows:

1. We build a cryptographic framework that combines the notion of Kolmogorov complexity and randomized Turing machines and use it to bound the minimum amount of bits required in order to evaluate a polynomial (Sect. 3).
2. We propose a formal definition of verifiable capacity-bound functions VCBFs that captures (a) a lower bound  $m$  on the number of bits read from memory (of bounded size) for evaluation (minimum capacity), (b) efficient verification of outputs with minimum capacity that is sublinear in  $m$  with respect to any malicious evaluator, and (c) soundness, i.e., no computationally bounded adversary can produce an incorrect output that passes verification (Sect. 4). We stress that the minimum capacity definition of VCBF (Sect. 4) significantly changes the perspective about how adversaries are usually modeled in cryptography. In our setting, the power of an adversary is solely dependent on the space it uses, i.e., the adversary has unbounded computational power, but it has limitations in the space it uses.<sup>3</sup> In a nutshell, an adversary is only limited to the size of available (precomputed) memory and the number of bits it reads from it. Considering space-only adversaries requires rethinking the meaning of adaptive security. As we will discuss next, adaptiveness refers to the ability of choosing the precomputation strategy (that sets the memory of the adversary) and the evaluation strategy (that sets the reading strategy during evaluation) after the VCBF's public parameters (i.e., the coefficients of the polynomial) are revealed. To work with such a space adaptive setting, Kolmogorov Complexity is essential and succeeds where any other standard entropy measure fails (see Sect. 1.1 for a detailed discussion).
3. We propose the first VCBF construction that satisfies our definition, based on single-variable polynomial evaluation for polynomial  $f(X) \in \mathbb{F}_p[x]$  of degree  $d$ . To achieve efficient verification, we employ the publicly verifiable and publicly delegatable verifiable computation scheme of [30]. For a target minimum capacity  $m \in O((d+1)\lambda)$ , it suffices to set the size of the polynomial to

---

<sup>3</sup> Considering unbounded adversaries is fundamental in order to capture the (concrete) strict lower bound on the number of distinct bits read that a VCBF must guarantee (i.e., a VCBF does not present any time/bits read trade-off). We provide a more detailed discussion in Sect. 4 and Remark 2.

$(d + 1)\lambda$ , where  $\lambda$  is the security parameter. Hence, to achieve large capacity bounds, we need to set  $d \gg \lambda$ , e.g.,  $d \in \Omega(\lambda^c)$  for  $c > 1$  constant. On the other hand the capacity complexity of the verification is  $O(\lambda)$ , i.e., independent of  $d$  hence verification remains efficient (Sect. 5).

4. In the full version of this work, we provide an estimation of the concrete parameters for our construction. For an elliptic curve group of order  $p$  of size 1024 bits, a polynomial of size 1 GB ( $d = 78.20 \cdot 10^5 \approx \lambda^{2.29}$ ) guarantees a minimum capacity  $m$  of 0.82 GB, even with respect to an unbounded adversary that can spend an exponential amount of computational resources.

We stress that a target minimum capacity  $m$  of a VCBF is guaranteed only in the presence of adversaries with a limited memory size  $n$ . As explained, the estimation of  $n$  is a major challenge when working in the standard setting (this work). Along this line, we initiate a fine-grained study on the memory size  $n$  estimation according to the number  $v$  of adaptive random accesses performed by the adversary (denoted by the set  $\mathcal{A}^{v\text{-access}}$ ). In particular, we prove (in the concrete setting) that the evaluation of a polynomial  $f(X) \in \mathbb{F}_p[x]$  guarantees a target capacity  $m \in O((d + 1)\lambda)$  even if an adversary  $A \in \mathcal{A}^{1\text{-access}}$  has access to a memory whose size  $n$  is proportional to the cardinality of the input space of the polynomial  $f(X)$ , i.e., super-polynomial. Our results can be extended to the asymptotic setting for the class  $\mathcal{A}^{O(1)\text{-access}}$  (Corollary 1). This result implies the security of our construction against adversarial strategies primarily used in practice (e.g., pre-computed dictionaries) or strategies executed on limited devices that have a bound on the number of random accesses (e.g., for energy efficiency) that they can perform. In Sect. 4 and Sect. 5.1, we discuss our results in more detail.

Regarding the larger class of adversaries  $\mathcal{A}^{\omega(1)\text{-access}}$ , the minimum capacity of our polynomial-based VCBF construction deteriorates when  $n$  gets closer to  $d^{1+\delta}\lambda^{1+o(1)}$  for a constant  $\delta > 0$ . This is due to the work of Kedlaya and Umans [39]: They shows how to build a data structure  $D$  of size at most  $d^{1+\delta}\lambda^{1+o(1)}$  (only from the coefficients of  $f(X) \in \mathbb{F}_p[x]$ ) that allows them to evaluate  $f(X)$  over any of the points. This evaluation requires reading a non-constant number of elements from  $D$  (using  $\omega(1)$  random accesses) whose total size is at most  $O(\log(d)^{s_1}\lambda^{s_2})$  for some positive  $s_1, s_2 \in O(1)$ . Hence, the plain evaluation of a polynomial of degree  $d$  can not guarantee a minimum capacity of  $m \in \omega(\log(d)^{s_1}\lambda^{s_2})$  when an adversary  $A \in \mathcal{A}^{\omega(1)\text{-access}}$  has access to a memory of size  $n$ , close to or greater than  $d^{1+\delta}\lambda^{1+o(1)}$  (see Sect. 5.1 for more details).

### 1.1 Adaptive Security and Kolmogorov Complexity (vs. Selective Security and Other Entropy Measures)

Here, we provide an answer to two natural questions about the meaning of adaptive security (in the space setting) and Kolmogorov complexity. These points significantly differentiate the techniques used in this work from previous ones.

**Adaptive Security in the Space Setting.** In the standard computational time cryptographic setting, adaptive security refers to the ability of an adversary of changing its behavior according to the scheme’s parameters with the objective of increasing its advantage in breaking the scheme’s security. An example is the adaptive CCA security of public encryption in which an adversary wants to increase its advantage in distinguishing between two encryptions by adaptively choosing both the two challenge messages and the next query for the decryption oracle after seeing the public key and the answers received from previous decryption queries. The natural question we pose is “What does adaptive security mean for the minimum capacity definition of VCBF?”. To give a concrete answer to this question, it is necessary to rethink the meaning of adaptive security against adversaries whose power is measured by solely considering the memory used/read (as done in this work). Jumping ahead, the minimum capacity of VCBF (Sect. 4) guarantees that an adversary needs to read at least  $m$  bits from its memory  $\tau$  (of size  $n$ ) when asked to correctly evaluate the function on a random point. This must hold even if the adversary is computationally unbounded, and it is allowed to generate/organize its memory  $\tau$  by precomputing the VCBF according to its parameters (i.e., polynomial). In such a setting, the objective of an adversary is to break the security of VCBF by minimizing the number of distinct bits  $m$  read from the precomputed memory. To achieve this, an adversary may think of changing its compression/precomputation strategy after the VCBF’s parameters (i.e., the polynomial coefficients) are revealed.<sup>4</sup> This is analogous to the CCA public key encryption example in which an adversary changes its two challenge messages after seeing the public key and the answers of the decryption oracle. To formally define the intuitive security of VCBF (i.e., a strict lower bound on the number  $m$  of distinct bits read), it is fundamental to cover adaptive space-based adversaries (as described above). Indeed, if an adversary can change its precomputation/compression strategy after seeing the VCBF’s parameters and reduce, for example,  $m$  by  $\log(\lambda)$  bits then the strict lower bound is not strict anymore. For this reason, the natural definition of minimum capacity (Definition 4) requires that the function remains secure for any possible space-based adversary sampled after the instantiation of VCBF, i.e., the precomputation and evaluation strategy (i.e., the memory and the bits read) of the adversary can depend on the VCBF itself. Such a model of adaptive security requires the usage of Kolmogorov complexity (see next).

**Why Kolmogorov complexity?** Conventional entropy measures (including Yao entropy that leverages the notion of compression and Shannon) consider the incompressibility of objects only on expectation. It implicitly means that the compression strategy does not depend on the object (i.e., our polynomial of our VCBF construction) sampled from a distribution. The typical example is on [35, Page 10] (quoting): “Consider the ensemble consisting of all binary strings of

<sup>4</sup> For example, a particular (hard to guess) compressible pattern may be revealed after the polynomial coefficients are chosen. Note that this may happen (with a certain probability) even if the polynomial is sampled at random.



length 9999999999999999. By Shannon’s measure, we require 9999999999999999 bits on the average to encode a string in such an ensemble. However, the string consisting of 9999999999999999 1’s can be encoded in about 55 bits by expressing 9999999999999999 in binary and adding the repeated pattern 1”. Note that the above argument applies also to the Rényi family of entropies (e.g., min-entropy).<sup>5</sup> The Kolmogorov complexity overcomes these limitations by considering the worst-case scenario: It measures incompressibility in an absolute sense, i.e., the compression strategy can depend on the object. Hence, lower-bounds derived through Kolmogorov complexity are universal, and they do not hold only on expectation. Also, quoting [58]: “The Kolmogorov complexity of an object is a form of absolute information of the individual object. This is not possible to do by C.E. Shannon’s information theory. Unlike Kolmogorov complexity, information theory is only concerned with the average information of a random source”.

In the VCBF setting, these concepts translate into adaptive vs. selective security. Kolmogorov complexity allows us to bound the minimum capacity of VCBF in the adaptive setting in which the adversarial compression/precomputation strategy can depend on the VCBF’s parameters (this mimics the adversarial behavior of changing strategy after the parameters are revealed). As already discussed, this is fundamental in order to have strict (universal) lower bound on the number of bits  $m$  that an adversary needs to read to evaluate a VCBF. Adaptive security remains unachievable if we consider standard entropy measures: This is because Information Theory studies the average information in objects, i.e., compression/precomputation strategies are fixed before the object (i.e., polynomial) is revealed/sampled. Hence, Kolmogorov complexity remains a fundamental tool in order to deal with space-based adaptive security and, in turn, to prove the security of our polynomial-based VCBF.

## 1.2 Applications of VCBF

Since VCBF can be seen as a space-analog of VDF, replacing minimum sequential steps with a minimum number of bits retrieved from memory, we believe they can find applications in various settings where memory usage needs to be enforced. In this direction, we describe how VCBF can be used as an *energy-consumption function* to achieve fairness among ASIC and CPU participants. We then briefly discuss other promising VCBF applications. We emphasize that the objective of this work is to lay the foundation for VCBFs, providing an initial study about publicly verifiable asymmetric memory/space hardness in the standard model. Naturally, depending on the application, ad-hoc properties and/or slightly different flavors of VCBF may be required, opening interesting directions for subsequent works.

---

<sup>5</sup> This can also be seen by observing that the Rényi family of entropies is equivalent to Shannon entropy when considering uniform distributions (as considered in this work, e.g., polynomial’s coefficients are sampled at random).

**Energy-Consumption Function.** Juels and Brainard [37] proposed client-puzzles as a solution to mitigate denial of service attacks (the concept of cryptographic puzzles can be traced back to Merkle’s key exchange [43] and Dwork and Naor’s pricing function [27]). The general idea of such puzzles is to associate a cost to each resource allocation request by requiring the client to complete a task before the server performs any expensive operation, thus making large-scale attacks infeasible. Classic client-puzzles [8, 20, 22, 37, 57] will force adversaries to consume certain CPU cycles as the cost for attacks. However, state-of-the-art hash engines [14, 54] could be  $200,000\times$  faster and  $40,000\times$  more energy-efficient than a state-of-art multi-core CPU. Hence, denial-of-service attacks may still be feasible for ASIC-equipped adversaries, even when such client puzzles are deployed as counter-mechanisms.

Motivated by this, we propose to replace CPU cycles with alternative resources, i.e., energy consumption using a VCBF. Classic ASIC-resistant methods follow the memory-hard function approach, i.e., ensuring that solving the puzzle “costs” much memory. In this manner, the cost of manufacturing an ASIC for puzzle solving would increase proportionally to the chip area devoted to memory. However, as argued in [54], memory hardness only partially solves the problem since it does not address the energy aspect of ASIC advantage. Indeed, energy consumption can be more important than the one-shot ASIC manufacturing cost since the corresponding cost (due to electricity consumption) keeps accumulating with time. Hence, a function with a strict lower bound on energy consumption, due to off-chip memory accesses enforced via VCBF, could fill in a critical but often overlooked gap in ASIC resistance.

Our VCBF can be used as an energy-consumption function in the following protocol between a server  $S$  and a client  $C$ :

- $C$  contacts server  $S$ , requesting permission to use some service such as establishing TLS connection [24] or accepting an email [27].
- $S$  returns a fresh challenge  $x$  to the client  $C$ .
- $C$  evaluates VCBF  $f$  on  $x$ , and returns the output and proof  $\pi$  to  $S$ .
- The server  $S$  verifies the correctness of  $f(x)$ . If the verification succeeds, it allows  $C$  to use the service.

Jumping ahead, from Theorem 7, we can easily find a set of parameters so that an adversary needs to invest a sufficiently large amount of energy in computing the function. Observe that the client is required to compute the VCBF  $f$  on a (honest) challenge  $x$  chosen by the server. Another option is to allow the client to choose multiple challenges on its own as is usually done in client-puzzles. In this case, it is fundamental that  $f$  can not be amortized, i.e., the puzzle’s total energy-cost increases proportionally to the number of parallel evaluations (on different challenges) of  $f$ . We stress that this work does not study amortization but this is not an inherent limitation of VCBF as a primitive. Non-amortizable VCBFs can be studied in future works.

The above protocol can be extended to blockchain systems that support smart contracts. For example, a client  $C$  may be required to evaluate the VCBF  $f$  on input  $x = H(s, t)$  in order to trigger the execution of a smart contract  $S$ .

Here, inside the hash function  $H$ , we place  $s$  which is the current state of the smart contract and  $t$  a counter used to randomize the challenge  $x = H(s, t)$  (e.g.,  $t$  is incremented after each invocation of the contract or after a block is mined).<sup>6</sup> In this way, an adversary is desisted from monopolizing the service offered by the smart contract  $S$  for specific malicious purposes. For instance, if the smart contract runs a decentralized auction system, the adversary will not be able to produce spamming bids to delay the acceptance of valid bids from competitors. We stress that efficient public verification is essential in blockchain systems since verifiers, that check the correctness of executions, have limited resources.

**Real-Time Services (and VCBF vs. VDF).** Although VCBF and VDF are both efficiently verifiable, there are applications in which VDFs can not be used, whereas VCBFs can. Consider a server  $S$  that offers a real-time service in which it is a requirement to receive requests within a precise time frame (e.g., within 1 minute). Clearly, using a VDF to block denial of service attacks is not an option since the time required to evaluate the VDF will delay all users' requests and affect the quality of the real-time service. A concrete example is an auction service: Bids must be received before the end of the auction or within a given time frame. Hence, VCBFs offer a unique solution in scenarios in which creating a delay is not acceptable.

**The Filecoin Network.** Protocol Labs is working on Filecoin [51], a blockchain-based decentralized storage system that has gathered much visibility in the last few years (it raised over \$250 million through an ICO in 2017). In Filecoin, miners earn coins by offering their storage to clients interested in storing and replicating files. The mining power in Filecoin is proportional to the active storage offered by a miner. Thanks to its public and capacity efficient verification, a VCBF can play an important role in improving *proof of useful space* (a fundamental primitive in the Filecoin protocol), i.e., a primitive that allows miners to prove that they are using a significant amount of space to store (multiple) files. In particular, Filecoin is interested in designing a proof of useful space in the cost model [44]. However, a common problem of proof of useful space constructions (e.g., [33]) is the possibility of trading space for computation: An evaluator may erase some data and reconstruct it on the fly when needed. A VCBF can tremendously improve proof of useful space by enforcing rationality during the computation when working in the cost model (e.g., by replacing the RO with a VCBF in graph-labeling based constructions). For example, the minimum capacity of VCBFs may increase the costs (e.g., energy consumption) of regeneration of the erased data. This encourages evaluators to store the data in its entirety. Also, the VCBF public verification does not introduce any additional cost to verifiers with minimal resources in terms of space and energy.

---

<sup>6</sup> The challenge  $x = H(s, t)$  has this format since smart contracts cannot generate secret randomness to sample a random challenge.

## 2 Preliminaries

**Notation.** We assume the reader to be familiar with standard cryptographic notation.

### 2.1 Publicly Verifiable Computation for Polynomial Evaluation

A publicly verifiable computation scheme (VC) for polynomial evaluation allows a client to outsource the computation of a polynomial  $f$  to an untrusted server. We are interested in VC schemes that are both *publicly delegatable* and *publicly verifiable*. The former allows any querier to submit input to the server, while the latter allows any verifier to check the computation's correctness. Formally, a VC scheme for a family of polynomials  $\mathcal{F}$  with input space  $\mathcal{X}$  is composed of the following algorithms:

- Setup**( $1^\lambda, f$ ): Upon input the security parameter  $1^\lambda$  and a polynomial  $f \in \mathcal{F}$ , the randomized setup algorithm returns the evaluation key  $\text{ek}_f$  and the verification key  $\text{vk}_f$  for the polynomial  $f$ .
- ProbGen**( $\text{vk}_f, x$ ): Upon input the verification key  $\text{vk}_f$  for a polynomial  $f \in \mathcal{F}$  and an input  $x \in \mathcal{X}$ , the deterministic problem generation algorithm outputs an encoding  $\sigma_x$  and the verification key  $\text{vk}_x$  for the input  $x$ .
- Compute**( $\text{ek}_f, \sigma_x$ ): Upon input the evaluation key  $\text{ek}_f$  for a polynomial  $f \in \mathcal{F}$  and an encoding  $\sigma_x$  for input  $x \in \mathcal{X}$ , the deterministic computation algorithm returns a value  $y$  and a proof  $\pi_y$ .<sup>7</sup>
- Verify**( $\text{vk}_x, y, \pi_y$ ): Upon input the verification key  $\text{vk}_x$  for an input  $x \in \mathcal{X}$ , a value  $y \in \mathcal{Y}$ , and a proof  $\pi_y$ , the deterministic verification algorithm returns a decisional bit  $b$ .

Correctness of a publicly VC scheme captures the fact that an honest execution of the computation to evaluate a polynomial  $f \in \mathcal{F}$  on input  $x \in \mathcal{X}$  produces the correct output  $y = f(x)$  along with a proof  $\pi_y$  that correctly verifies. As for security, a malicious evaluator cannot convince an honest verifier that  $y^* \neq f(x^*)$  is the correct evaluation of  $f(x^*)$  on an arbitrary input  $x^* \in \mathcal{X}$  (*soundness*). For the formal definitions, we refer the reader to [30].

In this work, we are interested in single-variable polynomials  $f(X) \in \mathbb{F}_p[x]$  of degree  $d$  of the form  $f(X) = \sum_{i=0}^d a_i \cdot x^i$ . An example of such a VC scheme has been proposed by Elkhiyaoui et al. [30]. It uses an asymmetric bilinear pairing  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  where  $\mathbb{G}_1, \mathbb{G}_2$ , and  $\mathbb{G}_T$  are groups of prime order  $p$ , and its security follows from the  $(d/2)$ -Strong Diffie-Hellman assumption ( $(d/2)$ -SDH).

VC schemes allow verifiers to check the computation's correctness more efficiently than the work required to evaluate the polynomial honestly. By leveraging the  $(d/2)$ -SDH assumption, the publicly VC scheme proposed in [30] yields a constant time  $O(1)$  verification. This gives to our VCBF an efficient capacity verification when using the VC scheme of Elkhiyaoui et al. [30] (see Sect. 5.1).

<sup>7</sup> We explicitly detached  $y$  from its proof  $\pi_y$ . Several works define the output of the computation algorithm **Compute** as a singleton  $\sigma_y$  (the encoding of the output  $y$ ) defined as  $\sigma_y = (y, \pi_y)$ .

## 2.2 Kolmogorov Complexity

The Kolmogorov complexity [40] aims to measure the complexity of objects in terms of the minimum amount of bits required to represent them. We say that  $(T, \alpha)$  is a (possibly inefficient) description of a string  $x \in \{0, 1\}^*$  (in terms of algorithmic complexity) if  $T(\alpha) = x$ . We can look at  $T$  as a decoding algorithm and  $\alpha \in \{0, 1\}^*$  as an encoding of  $x$ . The minimum amount of bits needed to represent a fixed bit string  $x$  is measured by the Kolmogorov complexity  $C_T(x)$ . In more detail, the Kolmogorov complexity  $C_T(x)$  of a bit string  $x \in \{0, 1\}^*$  with respect to a deterministic Turing machine  $T$  (called reference Turing machine) is defined as  $C_T(x) = \min_{\alpha \in \{0, 1\}^*} \{|\alpha| : T(\alpha) = x\}$ . Similarly, the conditional Kolmogorov complexity

measures the complexity of  $x$  given some auxiliary information  $y \in \{0, 1\}^*$ , i.e.,  $C_T(x|y) = \min_{\alpha \in \{0, 1\}^*} \{|\alpha| : T(\langle \alpha, y \rangle) = x\}$  where  $\langle a, b \rangle$

denotes the self-delimiting coding of strings  $a$  and  $b$ .<sup>8</sup> The above definitions of Kolmogorov complexity are known as *plain* Kolmogorov complexity. The name comes from the fact that no constraints are put on the input  $\alpha$  of the Turing machine  $T$ . Another type of complexity, called *prefix-free* Kolmogorov complexity [40, Sect. 3], focuses only on prefix-free programs, i.e., Turing machines that only take in input strings encoded in a prefix-free fashion. In this work, we focus on the plain version, and we refer the reader to [40, Sect. 3] for a more detailed discussion about the prefix-free version.

The definition of plain Kolmogorov complexity can be made independent from the reference Turing machine. Indeed, Turing machines are enumerable. The code of any Turing machine  $T$  can be interpreted as a binary string  $i$ .<sup>9</sup> Therefore, we can define a *universal* Turing machine  $U$  as  $U(i, \alpha) = T_i(\alpha)$ . In other words,  $U$  simulates all possible computations that Turing machines perform by taking in input  $\alpha \in \{0, 1\}^*$  and the code  $i$  of the  $i$ -th Turing machine  $T_i$  and executes the computation  $T_i(\alpha)$ . Based on this observation, it has been proved that the Kolmogorov complexity with respect to different Turing machines is invariant only up to a constant that depends on the reference Turing machine.

**Theorem 1 (Invariance Theorem [40, Theorem 2.1.1]).** *There is a universal deterministic Turing machine  $U$  such that for any deterministic Turing machine  $T$ , there is a constant  $c_T$  that only depends on  $T$ , such that for any string  $x, y \in \{0, 1\}^*$  we have  $C_U(x) \leq C_T(x) + c_T$ .<sup>10</sup>*

Since the choice of the reference Turing machine does not significantly change the Kolmogorov complexity of any string, we express the Kolmogorov complexity using the universal Turing machine  $U$  as a reference machine.

**Definition 1.** The Kolmogorov complexity of a string  $x$  is defined as  $C(x) \stackrel{\text{def}}{=} C_U(x)$  and  $C(x|y) \stackrel{\text{def}}{=} C_U(x|y)$  for the universal Turing machine  $U$ .

<sup>8</sup> As we will discuss later, Kolmogorov Complexity considers constant-size Turing machines. This requires the use of a self-delimiting code to encode multiple inputs.

<sup>9</sup> Note that not all binary strings are valid Turing machines.

<sup>10</sup> The constant  $c_T$  corresponds to the self-delimiting description of the Turing machine  $T$ .

It is fundamental to restrict the definition of Kolmogorov complexity to constant-size Turing machines in order to rule out any ambiguity. Indeed, as mentioned in [40, Sect. 2.1.4], by removing the size constraint of  $\mathbb{T}$ , it is possible to assign low complexity to any string by simply selecting a reference Turing machine with large complexity (i.e., hardcode the string into the code of the Turing machine). Still, the size constraint does not reduce the number of languages recognizable by a Turing machine. For example, it was shown the existence of a universal Turing machine with 15 states, 2 symbols, and 30 state-symbol product (transition function) [47, 60], with a polynomial slowdown of  $O(t^6)$ .

**String Incompressibility.** A crucial notion derived from the Kolmogorov complexity is the incompressibility of a string [40, Definition 2.2.1] with respect to unbounded deterministic Turing machines.

**Definition 2 (Deterministic  $c$ -incompressibility [40, Definition 2.2.1]).** A string  $x \in \{0, 1\}^*$  is  $c$ -DET-incompressible if  $C(x) \geq |x| - c$ .

We will refer to the above definition as *deterministic  $c$ -incompressibility* ( $c$ -DET-incompressibility in short) since it covers deterministic Turing machines, i.e., the reference Turing machine of the Kolmogorov complexity is deterministic.

The following theorem provides a lower-bound on the number of  $c$ -DET-incompressible elements in a given set  $\mathcal{X}$ .

**Theorem 2 ([40, Theorem 2.2.1]).** Let  $c \geq 0$  be a positive constant. For each  $y \in \{0, 1\}^*$ , every finite set  $\mathcal{X}$  of cardinality  $m$  has at least  $m(1 - 2^{-c}) + 1$  elements  $x \in \mathcal{X}$  such that  $C(x|y) \geq \log(m) - c$ .

By leveraging Theorem 2, we can easily calculate the probability of sampling a  $c$ -DET-incompressible string from  $\mathcal{X}$ . The proof is deferred to full version.

**Theorem 3.** Let  $\mathcal{X}$  be a finite set of cardinality  $m$ , then the following probability holds:  $\Pr[x \text{ is } c\text{-DET-incompressible} \mid x \leftarrow_s \mathcal{X}] \geq 1 - 2^{-c} + 1/m$ .

**String Incompressibility in the Randomized Setting.** In cryptography, we deal with randomized adversaries represented by randomized Turing machines. However, the  $c$ -DET-incompressibility only covers deterministic Turing machines since the reference Turing machine (used to measure the Kolmogorov complexity) is deterministic. Accordingly, we extend the notion of incompressibility to randomized Turing machines.

**Definition 3 (Randomized  $(c, \ell_{rnd})$ -incompressibility).** A string  $x \in \{0, 1\}^*$  is  $(c, \ell_{rnd})$ -RND-incompressible if for all constant-size unbounded randomized Turing machine  $\mathbb{T}$  with randomness space  $\{0, 1\}^{\ell_{rnd}}$ , for all  $r \in \{0, 1\}^{\ell_{rnd}}$ , and for all  $\alpha \in \{0, 1\}^{|x|-c-1}$ , we have  $\Pr[\mathbb{T}(\alpha; r) = x] = 0$ .

Naturally, there is an obvious connection between the two definitions of incompressibility. Indeed, the randomness of a randomized Turing machine can be seen as part of the input of a deterministic one. The following Theorem 4 reports the formal result, whose proof is deferred to full version.

**Theorem 4.** *Let  $x \in \{0, 1\}^*$  be a string. If  $x$  is  $c$ -DET-incompressible (Definition 2) then  $x$  is  $(c', \ell_{rnd})$ -RND-incompressible (Definition 3) where  $c' = c + \ell_{rnd} + 2 \log(\ell_{rnd}) + 1 + O(1)$ .*

The factor  $\ell_{rnd} + 2 \log(\ell_{rnd}) + 1$  is due to the need of using a self-delimiting  $\delta$ -encoding (Elias delta coding) to encode the randomness  $r \in \{0, 1\}^{\ell_{rnd}}$ . Also, the relation between the two incompressibility definitions is up to a constant  $O(1)$  because of the invariance theorem (Theorem 1), i.e., any equality holds up to a constant factor.

### 3 Kolmogorov-Bound for Polynomial Evaluation

At each evaluation, a VCBF scheme forces the evaluator to read at least  $m$  distinct bits from its main memory. To achieve this functionality, our construction leverages a single variable polynomial  $f(X) = \sum_{i=0}^d a_i \cdot x^i \in \mathbb{F}_p[x]$  of degree  $d$ . Intuitively, on receiving a challenge  $x \in \{0, 1\}^{\ell_{in}}$ , an honest evaluator needs to read the coefficients  $(a_0, \dots, a_d) \in \mathbb{F}_p^{d+1}$  that determine the polynomial  $f(X)$  in order to compute  $y = f(x)$ . In this case, we obtain the desired functionality by setting  $|(a_0, \dots, a_d)| \geq m$ . However, a malicious evaluator may find an alternative strategy to compute  $y = f(x)$  and read fewer than  $m$  bits. In this section, we prove the lower bound on the number of bits read during the polynomial evaluation by leveraging the Kolmogorov complexity. Next, we provide some examples of strategies a malicious evaluator could adopt:

1. Compress the coefficients  $(a_0, \dots, a_d)$  into a smaller string  $\alpha$ . In this way, the evaluator just needs to read  $\alpha$ , decompress it into  $(a_0, \dots, a_d)$ , and evaluate  $f(X)$  on the desired point  $x$ .
2. Precompute a dictionary  $T \stackrel{\text{def}}{=} (f(x_0), \dots, f(x_n))$  composed of the evaluation of  $f(X)$  on points  $(x_0, \dots, x_n)$ . By accessing  $T$ , the malicious evaluator can simply read and return  $y_i = f(x_i)$  if the challenge  $x_i$  is one of the precomputed points. In this case, the malicious evaluator reads only  $|y_i| \leq |p| < m$ .
3. Instead of storing  $(a_0, \dots, a_d)$ , the evaluator may choose to store  $d + 1$  arbitrary points  $(x_0, \dots, x_d)$ , the corresponding evaluations  $(f(x_0), \dots, f(x_d))$ , and the prime  $p$ . These pieces of information are enough to recover  $a$  via polynomial interpolation. As a result, if the expression of  $(f(x_0), \dots, f(x_d))$ , the points  $(x_0, \dots, x_d)$  and the prime  $p$  could be effectively compressed, the evaluator will read fewer bits than expected when evaluating the polynomial.

To estimate the bits that an adversary/algorithm needs to read to evaluate  $f(X)$  correctly, we built a bridge between the Kolmogorov complexity and polynomial evaluation. Our approach is based on two main observations.

First, any string  $a$  (of appropriate size) can be encoded into  $f(X) = \sum_{i=0}^d a_i \cdot x^i$  by setting its coefficients to different sub-portions of  $a$ . Let  $p$  be a prime of size  $\lambda + 1$  bits. We can interpret a string  $a \in \{0, 1\}^{(d+1)\lambda}$  as  $a = a_0 || \dots || a_d$  where  $a_i \in \mathbb{F}_p$  (i.e.,  $|a_i| \leq \lambda < |p|$ ) and use  $(a_0, \dots, a_d)$  as the coefficients of  $f(X)$ .

Second, if algorithm  $T$  is able to compute  $(f(x_0), \dots, f(x_d))$  taking in input a string  $\alpha$  and the challenge  $d$  points  $(x_0, \dots, x_d)$ , then  $(T, \langle \alpha, x_0, \dots, x_d \rangle)$  is

a valid description of  $(f(x_0), \dots, f(x_d))$ . As explained in Item 3, the tuples  $(f(x_0), \dots, f(x_d))$ ,  $(x_0, \dots, x_d)$ , and the prime  $p$ , are enough to reconstruct  $(a_0, \dots, a_d)$  (i.e., the prime's size  $\lambda + 1$  guarantees the encoding is injective).

By combining the above two observations, we can easily lower bound the size of  $\alpha$  with the Kolmogorov complexity  $C(a)$  of  $a$ . In more detail, consider a Turing machine  $T'$  that first executes  $T(\alpha, x_0, \dots, x_d)$  to compute  $f(x_0), \dots, f(x_d)$ , and then retrieves and return  $a$  via polynomial interpolation. This implies that  $(T', \langle p, \alpha, x_0, \dots, x_d \rangle)$  is a description of  $a$ . As a consequence, the size of  $\alpha$  (the string that  $T$  would read to compute  $(f(x_0), \dots, f(x_d))$ ) cannot be too small and must be related to the complexity  $C(a)$  of  $a$ . Below, we provide the formal result whose proof is included in the full version of this work.

**Theorem 5 (Kolmogorov-bound for (adaptive) Polynomial Evaluation).** *For any  $\lambda \in \mathbb{N}$ , let  $a \in \{0, 1\}^{(d+1)\lambda}$  be a binary string and  $p$  a prime of size  $\lambda + 1$ , respectively. Fix the polynomial  $f(X) = \sum_{i=0}^d a_i \cdot x^i \in \mathbb{F}_p[x]$  of degree  $d$  with input space  $\{0, 1\}^{\ell_{in}}$  where  $a = a_0 \parallel \dots \parallel a_d$  and  $a_i \in \mathbb{F}_p$  for  $i \in [d]$ . If  $a$  is  $(c', \ell_{rnd})$ -RND-incompressible (Definition 3), then for every constant-size randomized unbounded Turing machine  $T$  with randomness space  $\{0, 1\}^{\ell_{rnd}}$ , every  $\alpha \in \{0, 1\}^m$ , every  $r \in \{0, 1\}^{\ell_{rnd}}$ , and every tuple  $(x_0, \dots, x_d)$  such that  $\forall_{i \neq j} i, j \in \{0, \dots, d\}, x_i \neq x_j$  and  $x_i \in \{0, 1\}^{\ell_{in}}$ , we have*

$$\Pr[(f(x_0), \dots, f(x_d)) = T(\alpha, x_0, \dots, x_d; r)] = 0 \text{ where } m = (d + 1)(\lambda - \ell_{in} - 2 \log(\ell_{in} - 1) - c' - \lambda - 2 \log((d + 1)\lambda - c') - 2 \log(\lambda + 1) - 4.$$

An alternative way to interpret Theorem 5 is that any possible description  $(T, \alpha)$  of  $f(X)$  is bigger than the parameter  $m$  (defined in Theorem 5). Also, note that Theorem 5 presents a loss factor that is proportional to  $(d + 1)\ell_{in}$ . This because each of the  $d + 1$  points may be correlated with the coefficients of  $f(X)$  (i.e., each point  $x_i$  is equal to the first  $\ell_{in}$  bits of  $a_i$ ). The correlation may reduce the number of bits that must be read to compute the evaluations.

Lastly, we stress that Kolmogorov complexity permits us to prove Theorem 5 under the universal quantification of any  $d + 1$  evaluation points and any adversarial strategy (i.e., any memory  $\alpha$  and evaluation strategy  $T$ ) selected after the polynomial. This is essential in the adaptive space-based setting (Sect. 1.1) in which we want to estimate the size of information generated/read w.r.t. an arbitrary precomputation of the polynomial. Indeed, the precomputation adopted by an adversary may depend on both the polynomial and an arbitrary distribution of the evaluation points (e.g., dictionary attack). This aspect is fundamental to prove the *adaptive* security of our VCBF (see Sect. 4 and Sect. 5.1).

## 4 Definition of Verifiable Capacity-Bound Functions

A VCBF forces an evaluator to read at least  $m$  distinct bits from its main memory. Moreover, a VCBF does not permit to trade time for capacity, i.e., an evaluator is forced to read  $m$  distinct bits independently from its computational capabilities. As explained in [53], the number of off-chip memory accesses impacts



the energy consumption of the machine. If the cache’s size is significantly smaller than  $m$ , evaluating the function requires significant resources. However, on the (honest) receiver’s side, the validity of the computation can be verified efficiently in terms of capacity.

Formally, a VCBF scheme  $\Pi$  with input space  $\{0, 1\}^{\ell_{in}}$  is composed of the following polynomial-time algorithms:

**Setup**( $1^\lambda, 1^k$ ): Upon input the security parameter  $1^\lambda$  and the capacity parameter  $1^k$  (the *capacity parameter*  $1^k$  regulates the actual capacity cost, i.e., the number of bits read by the evaluator), the randomized setup algorithm returns the evaluation key  $\text{ek}$  and the verification key  $\text{vk}$ .

**Eval**( $\text{ek}, x$ ): Upon input the evaluation key  $\text{ek}$  and an input  $x \in \{0, 1\}^{\ell_{in}}$ , the deterministic evaluation algorithm returns the output  $y$  and a proof  $\pi$ . In the paper, we use the notation  $y = \text{Eval}(\text{ek}, x)$  (or simply  $\text{Eval}(\text{ek}, x)$ ) to denote solely the output  $y$ .

**Verify**( $\text{vk}, x, y, \pi$ ): Upon input the verification key  $\text{vk}$ , an input  $x \in \{0, 1\}^{\ell_{in}}$ , an output  $y$ , and a proof  $\pi$ , the deterministic verification algorithm returns a decisional bit  $b$ .

Intuitively, a VCBF scheme is correct if the output of an honest execution of the evaluation algorithm is accepted by the verification algorithm. In addition, a VCBF scheme should satisfy the following three basic properties: *minimum capacity*, *soundness* and *capacity efficient verification*.

**Adaptive Minimum Capacity.** The name captures the scheme’s lower-bound on the number of distinct bits  $m$  that must be fetched from the main memory to evaluate the function. In more detail, on input a random challenge  $x \leftarrow \{0, 1\}^{\ell_{in}}$ , the adversary  $A$  is asked to return the correct output  $y = \text{Eval}(\text{ek}, x)$  while reading at most  $m$  bits from its main memory. We assume the main memory of  $A$  is bounded since there is a strict relationship between the memory available and  $A$ ’s advantage  $\epsilon$ . Indeed, as discussed in Sect. 3, a viable adversarial strategy is to precompute a relatively large dictionary  $\tau = (\text{Eval}(\text{ek}, x_1), \dots, \text{Eval}(\text{ek}, x_n))$  (stored in the main memory) and return  $\text{Eval}(\text{ek}, x)$ , if  $x$  has been precomputed and included into  $\tau$ . A larger memory would allow the adversary to store more precomputed values  $\text{Eval}(\text{ek}, x_i)$ , thus increasing the probability of success.

More formally, let  $\tau \in \{0, 1\}^n$  and  $x \in \{0, 1\}^{\ell_{in}}$  be the binary string representing the memory of the adversary  $A$  and a challenge, respectively. We denote with  $\mathcal{I}_{A(\tau, x; r)} = \{i_1, i_2, \dots, i_{n'}\}_{n' \leq n}$  the ordered set of  $n'$  distinct indexes read by  $A$  during the computation of the output  $y = A(\tau, x; r)$  for the corresponding challenge  $x$  while having access to memory  $\tau$  and randomness  $r \in \{0, 1\}^{\ell_{rnd}}$ . Intuitively, on input the challenge  $x$  and randomness  $r$ , the adversary  $A$  fetches the binary string  $\tau_{x, r} = b_{i_1} || \dots || b_{i_{n'}}$  from  $\tau$  (where  $b_i$  represents the  $i$ -th bit of  $\tau$  and  $\mathcal{I}_{A(\tau, x; r)} = \{i_1, i_2, \dots, i_{n'}\}$ ) and then compute the output  $y$  using the knowledge of  $\tau_{x, r}$ ,  $x$ , and  $r$ .<sup>11</sup> A VCBF scheme is secure in the *adaptive setting* if for

<sup>11</sup> Observe that  $\tau_{x, r}$  can be fetched from  $\tau$  in an adaptive fashion according to the challenge  $x$  and randomness  $r$ .

any unbounded adversary sampled after VCBF's instantiation (i.e., execution of **Setup**) it is infeasible to compute the correct output  $\text{Eval}(\text{ek}, x) \neq y = \text{A}(\tau, x; r)$  when reading  $|\mathcal{I}_{\text{A}(\tau, x; r)}| = m$  bits.<sup>12</sup>

**Definition 4 ((Adaptive) Minimum Capacity of VCBF).** Fix the keys  $(\text{ek}, \text{vk}) \leftarrow_{\text{s}} \text{Setup}(1^\lambda, 1^k)$ . A VCBF scheme  $\Pi$  with input space  $\{0, 1\}^{\ell_{in}}$  satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity with respect to keys  $(\text{ek}, \text{vk})$  if for all constant-size unbounded randomized adversaries  $\text{A}$  with randomness space  $\{0, 1\}^{\ell_{rnd}}$  and for all  $\tau \in \{0, 1\}^n$ , we have:

$$\Pr[\text{Eval}(\text{ek}, x) = y \wedge |\mathcal{I}_{\text{A}(\tau, x; r)}| = m \mid x \leftarrow_{\text{s}} \{0, 1\}^{\ell_{in}}, y = \text{A}(\tau, x; r)] \leq \epsilon, \quad (1)$$

where  $r \leftarrow_{\text{s}} \{0, 1\}^{\ell_{rnd}}$ .

Informally, Definition 4 states that if a VCBF scheme  $\Pi$  satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity then the only way for an (exponential time) adversary  $\text{A}$  to increase its advantage  $\epsilon$  is to either read more than  $m$  distinct bits or have access to a memory larger than  $n$  bits (e.g., by storing in the memory  $\tau \in \{0, 1\}^n$  more pre-computed values). This guarantees the impossibility of trading time for capacity.

Note that the evaluator must return the correct output  $y = \text{Eval}(\text{ek}, x)$  and not a verifying proof  $\pi$ . The infeasibility of computing a verifying proof for a false output is defined by the soundness property (see next Definition 6). The choice of defining these two properties independently allows us to define them with respect to different settings, i.e., unbounded vs. computational adversaries. As mentioned above, defining adaptive minimum capacity in the unbounded setting is necessary to properly capture the absence of time/bits read trade-offs. See Remark 2 for more details.

Moreover, the definition captures the adaptive space-based setting described in Sect. 1.1. This is because the quantifiers of the security definition states that the VCBF remains secure for any memory  $\tau$  and adversary  $\text{A}$  both selected after the VCBF's instantiation (i.e., **Setup** algorithm). Intuitively, each  $\tau$  (resp.  $\text{A}$ ) represents an arbitrary precomputed memory (resp. arbitrary evaluation/reading strategy) that can depend on  $\text{ek}$  and  $\text{vk}$  (e.g., polynomial's coefficients).

*Relation between the memory size  $n$  and the advantage  $\epsilon$ .* Definition 4 is optimal in the sense that it does not put any constraint on the indexes  $\mathcal{I}_{\text{A}(\tau, x; r)}$  read by the adversary  $\text{A}$ . This means that  $\text{A}$  can arbitrarily access its memory. For example, it may perform multiple random accesses to the memory  $\tau$ , i.e., perform one or more conditional jumps into specific memory indexes to read different portions of the memory). Hence, one (or more) couple of progressive indexes  $\{i_j, i_{j+i}\} \subset \mathcal{I}_{\text{A}(\tau, x; r)}$  may be not consecutive (i.e.,  $|i_j - i_{j+1}| > 1$ ).

The optimality of Definition 4 appears to be the primary (apparently insurmountable) obstacle when trying to relate the memory size  $n$  and the advantage  $\epsilon$ . To retain an advantage  $\epsilon$ , an adversary  $\text{A}$  may choose to store (in the

<sup>12</sup> Without loss of generality, we assume the adversary reads exactly  $m$  bits since the higher the number of bits read, the higher the probability to compute the correct output  $y = \text{Eval}(\text{ek}, x)$ .

memory) a precomputed data structure which contains (possibly partial) pre-computed values, e.g., some evaluations  $y = \text{Eval}(\text{ek}, x_i)$  of a subset of inputs  $\mathcal{X} \subset \{0, 1\}^{\ell_{in}}$  (precomputed dictionary). However, the estimation of the memory size  $n$  (required to store the data structure) highly depends on what type of precomputation is performed (e.g., the entropy of the precomputed values, the algorithm used, etc.) and on the type of encoding and memory access strategy used by  $A$  when fetching the data from memory  $\tau$  to answer to an incoming challenge  $x$ . Unfortunately, this turned out to be a primary challenge when having block-box access to  $A$  and working in the standard setting (i.e., no oracles, no idealized functionalities, no ROM).

As a foundation paper of VCBF, we initiate a fine-grained study regarding the level of minimum capacity that can be achieved according to specific classes of adversaries. In particular, we provide a feasibility result showing (in the concrete setting) the meaningful relation between parameters  $\epsilon$  and  $n$  (using an information-theoretic approach) when dealing with the smaller class of adversaries  $\mathcal{A}^{1\text{-access}}$ . Such a class is composed by all the adversaries that perform exactly one (adaptive) random access to the memory  $\tau$ , i.e., on input the memory  $\tau \in \{0, 1\}^n$ , the challenge  $x \in \{0, 1\}^{\ell_{in}}$ , and randomness  $r \in \{0, 1\}^n$ , an adversary  $A \in \mathcal{A}^{1\text{-access}}$  adaptively jumps to an index  $i \in [n - m + 1]$  (memory location) and reads  $m$  consecutive indexes. Formally, when dealing with  $A \in \mathcal{A}^{1\text{-access}}$ , the indexes  $\mathcal{I}_{A(\tau, x; r)} = \{i_1, \dots, i_m\}$  read by  $A$  are consecutive, i.e.,  $i_j + 1 = i_{j+1}$  for  $j \in [m - 1]$ .<sup>13</sup> Observe that in  $\mathcal{A}^{1\text{-access}}$  we can identify several adversarial strategies used mainly in practice, e.g., precomputed dictionary attacks or any rainbow table technique that leverages a single adaptive random access.

As we will see during the security analysis of our construction (Sect. 5.1), by restricting the adversaries to the ones of the class  $\mathcal{A}^{1\text{-access}}$ , we can use a counting argument to concretely estimate the memory size  $n$  that an adversary  $A \in \mathcal{A}^{1\text{-access}}$  requires in order to retain a fixed advantage  $\epsilon$ . For completeness, we also include the results regarding the class  $\mathcal{A}^{v\text{-access}}$  for  $1 \leq v \leq m$ , i.e., adversaries that perform exactly  $v$  (adaptive) random access to the memory (observe that Definition 4 coincides with Definition 5 when  $\mathcal{A} = \bigcup_{i=1}^m \mathcal{A}^{i\text{-access}}$ ). However, due to the limited power of counting arguments, the memory size estimation  $n$  presents an exponential loss proportional to the number  $v$  of random accesses that  $A \in \mathcal{A}^{v\text{-access}}$  performs. In any case, this is enough to show that there exists a VCBF that satisfies  $(\text{negl}, O((d+1)\lambda), o((d+1)\lambda), \omega(\lambda^s))$ -min-capacity (in the asymptotic setting) with respect to the class of adversaries  $\mathcal{A}^{O(1)\text{-access}}$  for every positive constant  $s$ . Regarding  $\mathcal{A}^{\omega(1)\text{-access}}$ , the minimum capacity of our construction remains unclear. What we know is that the evaluation of a polynomial can not satisfy minimum capacity for  $\epsilon \in \text{negl}$  and  $m \in \omega(\log(d)^{s_1} \lambda^{s_2})$  (for some positive  $s_1, s_2 \in O(1)$ ) when  $n$  is close to or greater than  $d^{1+\delta} \lambda^{1+o(1)}$  (for a constant  $\delta > 0$ ) because of the efficient data structure for polynomial evaluation

<sup>13</sup> Without loss of generality, we assume that reading the first  $m$  bits of  $\tau$  requires the adversary to perform a random access to the first index of  $\tau$ .

of Kedlaya and Umans [39] (see Sect. 5.1). We now provide the formal security definition of minimum capacity with respect to a specific class of adversaries  $\mathcal{A}$ .

**Definition 5 ( $\mathcal{A}$ -class (adaptive) minimum capacity of VCBF).** A VCBF scheme  $\Pi$  with input space  $\{0, 1\}^{\ell_{in}}$  satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity with respect to the class of adversaries  $\mathcal{A}$  if  $\Pi$  satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity of Definition 4 where  $A$  is sampled from  $\mathcal{A}$ .

*Remark 1.* The Definitions 4 and 5 give robust guarantees in terms of capacity (according to the corresponding class of adversaries). For example, they consider unbounded adversaries and the minimum capacity must hold for every possible adversary  $A$  and memory  $\tau$  after the instantiation of the scheme (execution of the setup algorithm). This corresponds to the adaptive space-based security setting described in Sect. 1.1. Also, there is a more fundamental aspect to consider regarding Definitions 4 and 5: They do not rely on any heuristic assumptions, such as the Random Oracle (RO) or the Ideal Cipher [21], to measure the number of read bits. In fact, previous definitions of bandwidth-hard or memory-hard functions [2–6, 15, 19, 21, 54] do not directly measure the bits read by the evaluator. Instead, those models only calculate the number of the random oracle queries for each step. Therefore, the gap between RO queries and the actual number of bits read by the evaluator is artificially ignored in previous models. Finally, we stress that both RO and Ideal Cipher definitions neglect (and do not take into account) the adversary’s strategy in organizing and accessing specific portions of the memory: A fundamental aspect that needs to be considered when proving specific concrete memory bounds (in the standard model) for VCBFs.

**Soundness.** Soundness captures the infeasibility of convincing the verifier that  $y^* \neq \text{Eval}(\text{ek}, x)$  is the correct output of the computation. In more detail, it is infeasible for a malicious evaluator to compute a triple  $(x^*, y^*, \pi^*)$  that verifies successfully, but  $y^*$  is not the correct output of the computation. Soundness is also fundamental to enforce the  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity (Definitions 4 and 5) of a VCBF scheme. For example, if soundness does not hold, a malicious evaluator can deceive the verifier by returning a proof  $\pi^*$  and an output  $y^* \neq \text{Eval}(\text{ek}, x)$  such that  $\text{Verify}(\text{vk}, x, y^*, \pi^*) = 1$ . In this case, the energy consumption is not guaranteed since the value  $y^*$  is incorrect and may have been computed without fetching any bit from the main memory.

**Definition 6 (Soundness of VCBF).** A VCBF scheme  $\Pi$  with input space  $\{0, 1\}^{\ell_{in}}$  is  $(\epsilon)$ -sound if for all PPT adversary  $A$  we have:

$$\Pr \left[ \begin{array}{l} \text{Verify}(\text{vk}, x, y, \pi) = 1 \text{ and} \\ \text{Eval}(\text{ek}, x) \neq y \end{array} \middle| \begin{array}{l} (\text{ek}, \text{vk}) \leftarrow_s \text{Setup}(1^\lambda, 1^k) \\ (x, y, \pi) \leftarrow_s A(1^\lambda, \text{ek}, \text{vk}) \end{array} \right] \leq \epsilon.$$

*Remark 2 (On the combination of minimum capacity and soundness).* Formalizing adaptive minimum capacity (Definitions 4 and 5) and soundness (Definition 6) separately allows us to define these notions with respect to two distinct settings, i.e., unbounded adversaries vs. computational bounded adversaries. In

turn, minimum capacity with respect to unbounded adversaries is fundamental to capturing the (concrete) strict lower bound on the number of distinct bits guaranteed by a VCBF. This is because the unbounded setting guarantees that the lower bound must be satisfied independently of the running time of the adversary, i.e., trading time for bits read is infeasible. Observe that the computationally bounded version of minimum capacity does not guarantee the absence of a time/bits read trade-off. For example, a VCBF presenting an exponential trade-off (e.g., a PPT adversary can choose not to read a few bits at the cost of doubling its running time) may satisfy, asymptotically speaking, computational minimum capacity. However, the concrete lower bound would not be strict since the adversary can play with the gap allowed by the trade-off (this is not allowed when considering minimum capacity w.r.t unbounded adversaries as in our results). This is problematic when the VCBF is instantiated in practice since it makes the capacity bound less clear. For this reason, we chose to formalize these notions separately instead of being combined into a single one with respect to computationally bounded adversaries. Naturally, since we consider computational soundness, the final security of the VCBF holds only against computationally bounded adversaries (unless we drop the VCBF’s efficient verification). Still, we emphasize once again that unbounded minimum capacity is fundamental since it guarantees the absence of a trade-off with which the (computationally bounded) adversary could play with. Lastly, it may seem that another natural approach is to combine minimum capacity and soundness into a single definition that considers unbounded adversaries. Unfortunately, this is not possible since a VCBF that has a capacity efficient verification (see next Definition 7) cannot satisfy, at the same time, both minimum capacity and soundness with respect to unbounded adversaries (soundness with respect to unbounded adversaries is also known as perfect soundness, i.e., it does not exist a valid proof for a false statement/output). This is because an exponential adversary always exists that brute-forces all pairs of proofs and outputs until it finds the one that verifies. By leveraging perfect soundness, the adversary is guaranteed that the corresponding output is the correct VCBF’s evaluation. This attack only requires reading the VCBF’s verification key  $\mathbf{vk}$ , whose size must be sublinear in the VCBF’s minimum capacity  $m$ . This is required to satisfy capacity efficient verification (see next Definition 7).

**Capacity Efficient Verification.** The resource considered by VCBFs is the capacity since an evaluator is forced to read  $m$  distinct bits from its main memory. The verifier, on the other hand, should not have the same workload. For this reason, we require a VCBF scheme  $\Pi$  to be efficiently verifiable:

**Definition 7 (Capacity Efficient Verification of VCBF).** If  $\Pi$  satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity (either Definition 4 or Definition 5) then an honest

execution of the verification algorithm requires at most fetching  $o(m)$  bits from the memory (i.e., sublinear in  $m$ ).<sup>14</sup>

In particular, in this work, the capacity parameter is of the form  $m \in O((d+1)\lambda)$  where  $d$  is the degree of a polynomial  $f(X) = \sum_{i=0}^d a_i \cdot x^i \in \mathbb{F}_p[x]$  and  $\lambda + 1$  is the size of the prime  $p$ . As we will see, to reach high capacities (such as GB or even TB), for a fixed  $\lambda$  we will have to set  $d \in O(\lambda^c)$  for a constant  $c \geq 1$ . Nevertheless, the verification will be independent of  $d$  by leveraging the publicly VC scheme of Elkhiyaoui et al. [30]. Hence, we will obtain at least  $O(\lambda^{c+1})$  of min-capacity for the evaluation, and at most  $O(\lambda)$  of min-capacity for the verification (see Sect. 5.1).<sup>15</sup>

**On Energy Consumption.** A motivation for VCBFs is ASIC resistance. State-of-the-art hash engines [14, 54] could be  $200,000\times$  faster and  $40,000\times$  more energy efficient than multi-core CPUs. However, the energy consumption for off-chip memory accesses is similar for CPUs and ASICs [54]. If we assume the ASIC can hardcode only  $s$  bits, min-capacity guarantees that the ASIC will transfer at least  $m - s$  bits from the external memory during the evaluation. If the energy cost is  $u$  nJ per bit for external memory accesses, the evaluation of the VCBF costs at least  $u(m - s)$  nJ.

## 5 VCBF from VC for Polynomial Evaluation

In this section we show how to build a VCBF from VC for polynomial evaluation.

**Construction 1.** Let  $\mathcal{F}_{\lambda,d,p} = \{f_a(X) = \sum_{i=0}^d a_i \cdot x^i \bmod p\}_{a \in \{0,1\}^{(d+1)\lambda}}$  be an ensemble of polynomials where  $a = a_0 || \dots || a_d$ ,  $\lambda \in \mathbb{N}$ ,  $d \in \mathbb{N}$ , and  $p$  is a prime of  $\lambda + 1$  bits. Let  $\text{VC} = (\text{Setup}_{\text{VC}}, \text{ProbGen}_{\text{VC}}, \text{Compute}_{\text{VC}}, \text{Verify}_{\text{VC}})$  be a publicly VC scheme for the class  $\mathcal{F}_{\lambda,d,p}$ . We build a VCBF scheme with input space  $\{0, 1\}^{\ell_{\text{in}}}$  in the following way:

**Setup**( $1^\lambda, 1^k$ ): Without loss of generality, we assume  $k = (d + 1)\lambda$ . On input the security parameter  $1^\lambda$  and the capacity parameter  $1^k$ , the setup algorithm samples  $a_0 || \dots || a_d = a \leftarrow_{\$} \{0, 1\}^{(d+1)\lambda}$  where  $|a_i| = \lambda$  for  $i \in \{0, \dots, d\}$ . Then, it outputs the evaluation key  $\text{ek} = (\text{ek}_{f_a}, \text{vk}_{f_a})$  and the verification key  $\text{vk} = \text{vk}_{f_a}$  where  $(\text{ek}_{f_a}, \text{vk}_{f_a}) \leftarrow_{\$} \text{Setup}_{\text{VC}}(1^\lambda, f_a)$  and  $f_a \in \mathcal{F}_{\lambda,d,p}$ .

**Eval**( $\text{ek}, x$ ): On input the evaluation key  $\text{ek} = (\text{ek}_{f_a}, \text{vk}_{f_a})$  and an input  $x \in \{0, 1\}^{\ell_{\text{in}}}$ , the evaluation algorithm returns  $(y, \pi) = \text{Compute}_{\text{VC}}(\text{ek}_{f_a}, \sigma_x)$  where  $(\sigma_x, \text{vk}_x) = \text{ProbGen}_{\text{VC}}(\text{vk}_{f_a}, x)$ .

<sup>14</sup> Observe that  $|\text{vk}| + |x| + |y| + |\pi| \in o(m)$  (i.e.,  $\text{vk}, \pi, y, x$  are “succinct”) is necessary to obtain a capacity-efficient verification of  $o(m)$ . This is because  $\text{vk}, \pi, y, x$  are part of the verification algorithm **Verify** of VCBF.

<sup>15</sup> In the verification,  $O(\lambda)$  is for reading a constant number of group elements of order  $p$  of size at most  $\lambda + 1$ . In the evaluation,  $O((d+1)\lambda) = O(\lambda^{c+1})$  is for the  $d$  coefficients  $(a_0, \dots, a_d) \in \mathbb{F}_p^{d+1}$  of the polynomial  $f(X) \in \mathbb{F}_p[x]$ .

$\text{Verify}(\text{vk}, x, y, \pi)$ : On input the verification key  $\text{vk} = \text{vk}_{f_a}$ , an input  $x \in \{0, 1\}^{\ell_{in}}$ , an output  $y \in \mathcal{Y}$ , and a proof  $\pi$ , the verification algorithm returns  $b = \text{Verify}_{\text{VC}}(\text{vk}_x, y, \pi)$  where  $(\sigma_x, \text{vk}_x) = \text{ProbGen}_{\text{VC}}(\text{vk}_{f_a}, x)$ .

In this scheme, honest evaluators need to read at least  $k = (d+1)\lambda$  bits to load all the coefficients of the polynomial regardless of the cost of generating the proof  $\pi$ . Correctness follows directly from the correctness of the underlying schemes. For security and verification complexity, we establish the following results.

## 5.1 Security Analysis

The soundness is trivial. It simply follows from the  $(\epsilon)$ -soundness of VC (see [30] for the formal definition of soundness for VC).

**Theorem 6 (Soundness).** *If VC is  $(\epsilon)$ -sound, then the VCBF scheme  $\Pi$  of Construction 1 with input space  $\{0, 1\}^{\ell_{in}}$  is  $(\epsilon)$ -sound (Definition 6).*

Next, we show the level of minimum capacity that our VCBF scheme  $\Pi$  of Construction 1 satisfies with respect to the class of adversaries  $\mathcal{A}^{v\text{-access}}$  (Definition 5) for  $1 \leq v \leq m$ . This is formalized by Corollary 7 whose proof is deferred to full version. At high level, the proof is divided into two parts.

First, we prove that Construction 1 satisfies an alternative definition of minimum capacity dubbed *decomposed minimum capacity*. This definition is identical to Definition 4 except that the memory  $\tau$  is decomposed into  $n$  distinct strings  $(\tau_1, \dots, \tau_n)$  such that  $\tau_i \in \{0, 1\}^m$  for  $i \in [n]$  (intuitively, each  $\tau_i$  represents one possible string of length  $m$  that the adversary can read and interpret from its main memory, i.e.,  $(\tau_1, \dots, \tau_n)$  is the *decomposition* of the main memory). Then, the adversary succeeds if there exists  $i \in [n]$  such that  $y = A(\tau_i, x; r_i)$  where  $r_i \leftarrow_s \{0, 1\}^{\ell_{rnd}}$  and  $x \leftarrow_s \{0, 1\}^{\ell_{in}}$ . By leveraging Theorem 5, for each string  $\tau_i \in \{0, 1\}^m$ , the adversary can compute at most  $d$  distinct points  $x \in \{0, 1\}^{\ell_{in}}$  under the condition that the coefficients  $(a_0, \dots, a_d)$  of the polynomial  $f_a(X) \in \mathcal{F}_{\lambda, d, p}$  are RND-incompressible.<sup>16</sup>

Second, we show that any VCBF that satisfies decomposed minimum capacity w.r.t.  $n - m + 1$  strings  $(\tau_1, \dots, \tau_{n-m+1})$  (each of length  $m$ ), also satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity (the standard definition) with respect to the class of adversaries  $\mathcal{A}^{1\text{-access}}$  (Definition 5). The result follows by using a counting argument: An adversary  $A \in \mathcal{A}^{1\text{-access}}$  with access to memory  $\tau$  of length  $n$  can read at most  $n - m + 1$  different strings each of length  $m$ . This argument can be generalized for each class  $\mathcal{A}^{v\text{-access}}$  for  $1 \leq v \leq m$ . Unfortunately, due to the limited power of counting arguments, the memory size  $n$  presents an exponential loss proportional to  $v$ .

**Theorem 7 ( $\mathcal{A}^{v\text{-access}}$ -class (adaptive) minimum capacity).** *Let  $v \in \mathbb{N}$  and  $\Pi$  be a VCBF scheme with input space  $\{0, 1\}^{\ell_{in}}$ . Fix the keys*

<sup>16</sup> Note that the polynomial  $f_a(X)$  is RND-incompressible with overwhelming probability since it is sampled at random. This follows by leveraging Theorems 3 and 4.

$(\mathbf{ek}, \mathbf{vk}) \leftarrow_s \text{Setup}(1^\lambda, 1^k)$ . The VCBF scheme II of Construction 1 with input space  $\{0, 1\}^{\ell_{in}}$  satisfies  $(\epsilon, m, \ell_{rnd}, n)$ -min-capacity with respect to the class of adversaries  $\mathcal{A}^{v\text{-access}}$  and keys  $(\mathbf{ek}, \mathbf{vk})$  (Definition 5) where  $\lambda \in \mathbb{N}, d \in \mathbb{N}, c \in \mathbb{N}, \epsilon_1 \in [0, 1]$ ,

$$\begin{aligned} m &= (d+1)(\lambda - \ell_{in} - 2\log(\ell_{in}) - 1) - c' \\ &\quad - \lambda - 2\log((d+1)\lambda - c') - 2\log(\lambda + 1) - 4, \\ c' &= c + \ell_{rnd} + 2\log(\ell_{rnd}) + 1 + O(1), \\ \epsilon &= \epsilon_1 + \frac{d+1}{2^{\ell_{in}}} + \frac{1}{2^c} - \frac{1}{2^{(d+1)\lambda}}, \\ n &= \begin{cases} m + \frac{\epsilon_1 \cdot 2^{\ell_{in}}}{d} & \text{if } v = 1 \\ v \sqrt{\left(\frac{\epsilon_1 \cdot 2^{\ell_{in}}}{d} + 1\right) / \left(v! \left(\frac{m-1}{v-1}\right)^{v-1}\right)} \cdot v & \text{if } 1 < v \leq m. \end{cases} \end{aligned}$$

Recall that in the class of adversaries  $\mathcal{A}^{1\text{-access}}$  we find common adversarial strategies (primarily used in practice) such as precomputed dictionary attacks (e.g., ordered dictionary in which the  $x$ -th evaluation  $f(x)$  is stored at the  $x$ -th offset) or limited devices that are hindered from performing non-constant random accesses (e.g., for energy efficiency). Also, we stress that, if we consider memories of size  $n = m$ , our Construction 1 satisfies  $(\epsilon, m, \ell_{rnd}, m)$ -min-capacity with respect to the *optimal* Definition 4 (i.e., security against adversaries that arbitrarily access its memory) where  $\epsilon = \frac{d+1}{2^{\ell_{in}}} + \frac{1}{2^c} - \frac{1}{2^{(d+1)\lambda}}$ . This because  $\tau \in \{0, 1\}^m$  only allows an adversary to answer to at most  $d$  points (Theorem 5).

The following asymptotic Corollary 1 shows that a secure VCBF exists (in the standard model) with respect to the class of adversary  $\mathcal{A}^{O(1)\text{-access}}$ . We stress that this must be interpreted as a purely theoretical result showing the feasibility of VCBF since the constants hidden by the asymptotic notation are large.

**Corollary 1.** *For any  $\lambda \in \mathbb{N}$  and  $k = (d+1)\lambda \in \mathbb{N}$  such that  $d \in \mathbb{N}$ , there exists a VCBF that satisfies  $(\text{negl}, O((d+1)\lambda), o((d+1)\lambda), \omega(\lambda^s))$ -min-capacity with respect to the class of adversaries  $\mathcal{A}^{O(1)\text{-access}}$  for every constant  $s \geq 1$ .<sup>17</sup>*

**Verification Complexity.** Corollary 1 shows that an evaluator needs to read at least  $O((d+1)\lambda)$  distinct bits from its main memory. We now analyze the verifier capacity complexity. By inspecting Construction 1, we observe that the capacity complexity of `Verify` coincides with the ones of algorithms `ProbGenVC` and `VerifyVC` of the underlying VC scheme. Therefore, we must consider a concrete instantiation of the VC scheme. For this reason, we measured the efficiency of our VCBF with respect to the VC scheme of Elkhyaoui et

<sup>17</sup> We stress that the memory size  $n$  does not need to be super-polynomial (in the security parameter) in order to consider a VCBF secure. Indeed, in a scenario in which a machine has at most  $n = \lambda^s \in \text{poly}$  bits of free memory (for a positive constant  $s$ ), it is enough to show that the VCBF satisfies  $(\epsilon, m, \ell_{rnd}, \lambda^s)$ -min-capacity where  $\epsilon$  is the target advantage.



al. [30] that uses an asymmetric bilinear pairing  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  in which the  $(d/2)$ -SDH assumption holds. The execution of  $\text{ProbGen}_{\text{VC}}(\text{vk}_f, x)$  computes and returns  $\text{vk}_x = (\text{vk}_x^0, \text{vk}_x^1) = (g_{b_0} \cdot g^{x^2}, h_{r_1}^x \cdot h_{r_0})$  and  $\sigma_x = x$ , where  $\text{vk}_f = (g_{b_0}, h_{r_1}, h_{r_0}) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_2$ ,  $x \in \mathbb{F}_p$ , and  $g$  the generator of  $\mathbb{G}_1$  (observe that the size of the verification key  $\text{vk}_f$  is  $O(\lambda)$ , i.e., does not depend on the degree  $d$  of the polynomial). Moreover,  $\text{Verify}_{\text{VC}}(\text{vk}_x, y, \pi_y)$  verifies the correctness of the computation by checking the equality  $e(g, h^y) \stackrel{?}{=} e(\text{vk}_x^0, \pi_y) \cdot e(g, \text{vk}_x^1)$ , where  $\text{vk}_x = (\text{vk}_x^0, \text{vk}_x^1)$  and  $h$  is a generator of  $\mathbb{G}_2$ . Hence, in the worst case, the verification capacity complexity of our VCBF is  $O(\lambda) \in o((d+1)\lambda)$ , while the verification time is  $O(1)$  in the number of group operations. This is because the executions of  $\text{ProbGen}_{\text{VC}}$ ,  $\text{Verify}_{\text{VC}}$ , and the sizes of  $(\text{vk}_f, x, y, \pi_y)$  (that compose the inputs of  $\text{ProbGen}_{\text{VC}}$  and  $\text{Verify}_{\text{VC}}$ ), are independent of the polynomial degree  $d$  in terms of both capacity and time.

**Improve the Memory Size Bound.** For  $v \in \omega(1)$ , our VCBF construction needs to face the efficient data structure for polynomial evaluation of Kedlaya and Umans [39]. In particular, they show that, for any constant  $\delta > 0$ , there exists a data structure  $D$  of size  $d^{1+\delta} \lambda^{1+o(1)}$  that can be computed by preprocessing only the coefficients of  $f(X) \in \mathbb{F}_p[X]$ . An evaluator in  $\mathcal{A}^{\omega(1)\text{-access}}$  can correctly evaluate  $f(x)$  on every  $x \in \{0, 1\}^{\ell_{in}}$  in time  $\text{polylog}(d) \cdot \lambda^{1+o(1)}$ , performing a non-constant number of random accesses and reading at most  $\text{polylog}(d) \lambda^{1+o(1)} \cdot w$  bits from  $D$  (with  $w \in O(\lambda)$  we denote bit size of the elements contained in  $D$ ). Hence, our VCBF construction can not achieve  $(\epsilon, m, \ell_{rnd}, n)$  for  $\epsilon \in \text{negl}$  and  $m \in \omega(\log(d)^{s_1} \lambda^{s_2})$  (for some positive  $s_1, s_2 \in O(1)$ ) when  $n$  is close to or greater than  $d^{1+\delta} \lambda^{1+o(1)}$ . The above observation poses the natural question of whether an asymptotic VCBF (in the  $\mathcal{A}^{\omega(1)\text{-access}}$  setting) that satisfies min-capacity for reasonably large  $m$  and  $n$  super-polynomial in  $\lambda$  as in Corollary 1 (i.e.,  $n$  asymptotically larger than the size  $d^{1+\delta} \cdot \lambda^{1+o(1)}$  of the data structure of Kedlaya and Umans [39]). The answer to the important question requires a non-trivial and precise study that can be undertake in future works.

**Acknowledgments.** We thank Irene Giacomelli and Luca Nizzardo for helpful discussions.

The authors were partially supported by Protocol Labs under the RFP-009 on Proof of Space and Useful Space. In addition, the second author was supported by the National Key R&D Program of China 2021YFB3100100 and CAS Project for Young Scientists in Basic Research Grant YSBR-035, the third author was supported by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM), and the fourth author was supported by Hong Kong Research Grants Council under grant GRF-16200721.

## References

1. Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. *ACM Trans. Internet Technol. (TOIT)* **5**(2), 299–327 (2005)

2. Alwen, J., Blocki, J.: Efficiently computing data-independent memory-hard functions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 241–271. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53008-5\\_9](https://doi.org/10.1007/978-3-662-53008-5_9)
3. Alwen, J., Blocki, J., Harsha, B.: Practical graphs for optimal side-channel resistant memory-hard functions. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1001–1017 (2017)
4. Alwen, J., Blocki, J., Pietrzak, K.: Sustained space complexity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 99–130. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78375-8\\_4](https://doi.org/10.1007/978-3-319-78375-8_4)
5. Alwen, J., Chen, B., Pietrzak, K., Reyzin, L., Tessaro, S.: **Scrypt** is maximally memory-hard. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 33–62. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_2](https://doi.org/10.1007/978-3-319-56617-7_2)
6. Alwen, J., Serbinenko, V.: High parallel complexity graphs and memory-hard functions. In: Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing, pp. 595–603 (2015)
7. Ateniese, G., Bonacina, I., Faonio, A., Galesi, N.: Proofs of space: when space is of the essence. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 538–557. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10879-7\\_31](https://doi.org/10.1007/978-3-319-10879-7_31)
8. Aura, T.: DOS-resistant authentication with client puzzles. In: Christianson, B., Malcolm, J.A., Crispo, B., Roe, M. (eds.) Security Protocols 2000. LNCS, vol. 2133, pp. 178–181. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44810-1\\_23](https://doi.org/10.1007/3-540-44810-1_23)
9. Back, A.: Hashcash—a denial of service counter-measure (2002)
10. Bellare, M., Kane, D., Rogaway, P.: Big-key symmetric encryption: resisting key exfiltration. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 373–402. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53018-4\\_14](https://doi.org/10.1007/978-3-662-53018-4_14)
11. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: Black-Box, White-Box, and Public-Key (Extended Abstract). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 63–84. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-45611-8\\_4](https://doi.org/10.1007/978-3-662-45611-8_4)
12. Biryukov, A., Khovratovich, D.: Egalitarian computing. In: Holz, T., Savage, S. (eds.) USENIX Security 2016, pp. 315–326. USENIX Association, August 2016
13. Biryukov, A., Perrin, L.: Symmetrically and asymmetrically hard cryptography. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 417–445. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70700-6\\_15](https://doi.org/10.1007/978-3-319-70700-6_15)
14. Bitmain: Antminer s9 (2020). <https://shop.bitmain.com/product/detail?pid=00020200306153650096S2W5mYli0661>
15. Blocki, J., Ren, L., Zhou, S.: Bandwidth-hard functions: reductions and lower bounds. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1820–1836 (2018)
16. Bogdanov, A., Isobe, T.: White-box cryptography revisited: space-hard ciphers. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1058–1069 (2015)
17. Bogdanov, A., Isobe, T., Tischhauser, E.: Towards practical whitebox cryptography: optimizing efficiency and space hardness. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 126–158. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_5](https://doi.org/10.1007/978-3-662-53887-6_5)

18. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-96884-1\\_25](https://doi.org/10.1007/978-3-319-96884-1_25)
19. Boneh, D., Corrigan-Gibbs, H., Schechter, S.E.: Balloon hashing: a memory-hard function providing provable protection against sequential attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 220–248. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_8](https://doi.org/10.1007/978-3-662-53887-6_8)
20. Canetti, R., Halevi, S., Steiner, M.: Hardness amplification of weakly verifiable puzzles. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 17–33. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-30576-7\\_2](https://doi.org/10.1007/978-3-540-30576-7_2)
21. Chen, B., Tessaro, S.: Memory-hard functions from cryptographic primitives. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 543–572. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26951-7\\_19](https://doi.org/10.1007/978-3-030-26951-7_19)
22. Chen, L., Morrissey, P., Smart, N.P., Warinschi, B.: Security notions and generic constructions for client puzzles. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 505–523. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10366-7\\_30](https://doi.org/10.1007/978-3-642-10366-7_30)
23. Cohen, B., Pietrzak, K.: Simple proofs of sequential work. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 451–467. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78375-8\\_15](https://doi.org/10.1007/978-3-319-78375-8_15)
24. Dean, D., Stubblefield, A.: Using client puzzles to protect TLS. In: USENIX Security Symposium, vol. 42 (2001)
25. Döttling, N., Lai, R.W.F., Malavolta, G.: Incremental proofs of sequential work. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 292–323. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17656-3\\_11](https://doi.org/10.1007/978-3-030-17656-3_11)
26. Dwork, C., Goldberg, A., Naor, M.: On memory-bound functions for fighting spam. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 426–444. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45146-4\\_25](https://doi.org/10.1007/978-3-540-45146-4_25)
27. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). [https://doi.org/10.1007/3-540-48071-4\\_10](https://doi.org/10.1007/3-540-48071-4_10)
28. Dwork, C., Naor, M., Wee, H.: Pebbling and proofs of work. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 37–54. Springer, Heidelberg (2005). [https://doi.org/10.1007/11535218\\_3](https://doi.org/10.1007/11535218_3)
29. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 585–605. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48000-7\\_29](https://doi.org/10.1007/978-3-662-48000-7_29)
30. Elkhiyaoui, K., Önen, M., Azraoui, M., Molva, R.: Efficient techniques for publicly verifiable delegation of computation. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 119–128. ACM (2016)
31. 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, Heidelberg (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_25](https://doi.org/10.1007/978-3-030-45721-1_25)
32. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012, pp. 501–512. ACM Press, October 2012
33. Fisch, B.: Tight proofs of space and replication. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 324–348. Springer, Heidelberg (May (2019))

34. Fouque, P.-A., Karpman, P., Kirchner, P., Minaud, B.: Efficient and provable white-box primitives. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 159–188. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_6](https://doi.org/10.1007/978-3-662-53887-6_6)
35. Grunwald, P., Vitányi, P.: Shannon information and Kolmogorov complexity. arXiv preprint cs/0410002 (2004)
36. Jaeger, J., Tessaro, S.: Tight time-memory trade-offs for symmetric encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 467–497. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17653-2\\_16](https://doi.org/10.1007/978-3-030-17653-2_16)
37. Juels, A.: Client puzzles: a cryptographic countermeasure against connection depletion attacks. In: Proceedings of Networks and Distributed System Security Symposium (NDSS) (1999)
38. Kaliski, B.: Password-based cryptography specification. RFC 2898 (2000)
39. Kedlaya, K.S., Umans, C.: Fast modular composition in any characteristic. In: 2008 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 146–155. IEEE (2008)
40. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. TCS. Springer, New York (2008). <https://doi.org/10.1007/978-0-387-49820-1>
41. Liu, Y., Pass, R.: On one-way functions and Kolmogorov complexity. In: FOCS 2020, 61st Annual IEEE Symposium on Foundations of Computer Science (2020)
42. Mahmoody, M., Moran, T., Vadhan, S.: Publicly verifiable proofs of sequential work. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, pp. 373–388 (2013)
43. Merkle, R.C.: Secure communications over insecure channels. Commun. ACM **21**(4), 294–299 (1978)
44. Moran, T., Orlov, I.: Simple proofs of space-time and rational proofs of storage. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 381–409. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26948-7\\_14](https://doi.org/10.1007/978-3-030-26948-7_14)
45. Muchnik, A.A.: Kolmogorov complexity and cryptography. Proc. Steklov Inst. Math. **274**(1), 193 (2011)
46. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
47. Neary, T., Woods, D.: Four small universal Turing machines. Fundamenta Informaticae **91**(1), 123–144 (2009)
48. 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](https://doi.org/10.1007/978-3-642-36594-2_13)
49. Percival, C.: Stronger key derivation via sequential memory-hard functions (2009)
50. Pietrzak, K.: Simple verifiable delay functions. In: 10th Innovations in Theoretical Computer Science Conference (ITCS 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
51. Protocol Labs: Filecoin: a decentralized storage network (2017). <https://filecoin.io/filecoin.pdf>. Accessed 8 Apr 2023
52. Provos, N., Mazieres, D.: A future-adaptable password scheme. In: USENIX Annual Technical Conference, FREENIX Track, pp. 81–91 (1999)
53. Ren, L., Devadas, S.: Proof of space from stacked expanders. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9985, pp. 262–285. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53641-4\\_11](https://doi.org/10.1007/978-3-662-53641-4_11)
54. Ren, L., Devadas, S.: Bandwidth hard functions for ASIC resistance. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 466–492. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70500-2\\_16](https://doi.org/10.1007/978-3-319-70500-2_16)

55. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto (1996)
56. Souto, A., Teixeira, A., Pinto, A.: One-way functions using Kolmogorov complexity. In: Proceedings of the Computability in Europe, pp. 346–356 (2010)
57. Stebila, D., Kuppusamy, L., Rangasamy, J., Boyd, C., Gonzalez Nieto, J.: Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 284–301. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19074-2\\_19](https://doi.org/10.1007/978-3-642-19074-2_19)
58. Vitányi, P.: Personal webpage. <https://homepages.cwi.nl/paulv/kolmogorov.html>
59. Wesolowski, B.: Efficient verifiable delay functions. *J. Cryptol.* 1–35 (2020)
60. Woods, D., Neary, T.: The complexity of small universal Turing machines: a survey. *Theor. Comput. Sci.* **410**(4–5), 443–450 (2009)
61. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: verifying arbitrary SQL queries over dynamic outsourced databases. In: 2017 IEEE Symposium on Security and Privacy, pp. 863–880. IEEE Computer Society Press, May 2017



# A Holistic Approach Towards Side-Channel Secure Fixed-Weight Polynomial Sampling

Markus Krausz<sup>1</sup>, Georg Land<sup>1,2</sup>(✉), Jan Richter-Brockmann<sup>1</sup>,  
and Tim Güneysu<sup>1,2</sup>

<sup>1</sup> Horst Görtz Institute for IT Security, Ruhr University Bochum, Bochum, Germany  
{markus.krausz,georg.land,jan.richter-brockmann,tim.gueysu}@rub.de

<sup>2</sup> Cyber-Physical Systems, DFKI GmbH, Bremen, Germany

**Abstract.** The sampling of polynomials with fixed weight is a procedure required by round-4 Key Encapsulation Mechanisms (KEMs) for Post-Quantum Cryptography (PQC) standardization (BIKE, HQC, McEliece) as well as NTRU, Streamlined NTRU Prime, and NTRU LPRime. Recent attacks have shown in this context that side-channel leakage of sampling methods can be exploited for key recoveries. While countermeasures regarding such timing attacks have already been presented, still, there is no comprehensive work covering solutions that are also secure against power side channels.

To close this gap, the contribution of this work is threefold: First, we analyze requirements for the different use cases of fixed weight sampling. Second, we demonstrate how *all* known sampling methods can be implemented securely against timing and power/EM side channels and propose performance-enhancing modifications. Furthermore, we propose a new, comparison-based methodology that outperforms existing methods in the masked setting for the three round-4 KEMs BIKE, HQC, and McEliece. Third, we present bitsliced and arbitrary-order masked software implementations and benchmarked them for all relevant cryptographic schemes to be able to infer recommendations for each use case. Additionally, we provide a hardware implementation of our new method as a case study and analyze the feasibility of implementing the other approaches in hardware.

**Keywords:** PQC · Fixed Weight Polynomial Sampling · Higher-order Masking · Cortex-M4

## 1 Introduction

With the potential advent of large-scale quantum computers, rendering “classic” asymmetric cryptosystems like Elliptic Curve Cryptography (ECC) insecure, wide deployment of Post-Quantum Cryptography (PQC) has become inevitable. After three rounds of thorough analysis and many broken cryptosystems, a first set of algorithms has been selected for standardization. To enable further diversification

of security assumptions, a fourth round of standardization has been launched, consisting of the three code-based schemes BIKE, HQC, and McEliece.

One building block for all round-four candidates is fixed-weight polynomial sampling. Additionally, this is also required in the three lattice-based schemes NTRU, which may replace Kyber if potential patent issues are not resolved, Streamlined NTRU Prime, which is currently the default algorithm in OpenSSH 9, and NTRU LPRime. The output of this sampling is a uniform random binary or ternary polynomial of a specific size with a fixed number of non-zero coefficients. Multiple algorithmic approaches have been proposed [5, 9, 10, 15, 18] for this.

Karabulut et al. presented the first power side-channel attack on fixed weight sampling [14], targeting NTRU, Streamlined NTRU Prime, and Dilithium. Recently, Guo et al. [12] introduced an attack on HQC and BIKE utilizing the fixed weight polynomial sampling with variable timing depending on the seed. Sendrier [18] seized their approach and presented suitable countermeasures for BIKE. While this attack exploits timing differences, there is no reason to believe that a power side channel cannot be exploited analogously.

On the defense end, however, there is no comprehensive analysis of effective countermeasures against this type of attack. In particular, given these recent attacks, it becomes urgent to develop also power side-channel secure methodologies for fixed-weight polynomial sampling.

Hence, we present a holistic examination of the fixed-weight polynomial sampling problem with different attacker models, parameters, sampling methods, and implementation variants. We show how power side-channel secure variants of all suitable algorithms can be realized, propose performance-enhancing modifications, and provide bitsliced masked software implementations for arbitrary masking order which we make publicly available<sup>1</sup>. Additionally, we develop a new probabilistic sampling method accompanied by a hardware implementation and a new methodology for Boolean masked comparison which is a core component for multiple algorithms. We benchmark and evaluate our implementations for all relevant PQC schemes.

## 2 Preliminaries

The two most important parameters for the fixed-weight polynomial sampling problem are the length of the polynomial and the weight (number of non-zero coefficients) denoted by  $N$  and  $W$  throughout the paper.

**Binomial Distribution.** For the Binomial probability distribution, we denote the probability mass function as

$$\mathcal{B}(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k} \quad (1)$$

where  $k$  is the number of successes in  $n$  independent Bernoulli trials, each with probability  $p$ . We know that  $\mathcal{B}(k, n, p)^{-1}$  is the expected number of repetitions of the overall experiment until *exactly*  $k$  out of  $n$  successes are reached.

<sup>1</sup> <https://github.com/Chair-for-Security-Engineering/maskedFWPS>.

## 2.1 Side-Channel Analysis

In this work, we consider timing behavior and power consumption of a target implementation of a cryptographic algorithm as possible side channels that could be exploited by an attacker. For timing attacks, we consider runtime differences caused by memory or cache accesses, branching on sensitive data, or secret-dependent arithmetic operations.

For power side-channel attacks, we distinguish between single-trace and multi-trace attacks. In the single-trace scenario, the attacker has given only one single trace of the cryptographic operation, i.e., the attacker cannot invoke the system multiple times with the same secret key. However, we additionally assume that an attacker can mount template attacks. In this case, the attack profiles a target device to create a power template which is used to match a single trace to the correct key.

For multi-trace attacks, we assume that an attacker can collect as many traces as possible. These traces are used for Differential Power Analysis (DPA) including statistical analyses like Correlation Power Analysis (CPA).

## 2.2 Masking

Masking is a well-established countermeasure against physical Side-Channel Analysis (SCA) and is based on the strong theoretic foundation of secret sharing. A secret value  $x$  is split into  $d + 1$  shares  $x_i$  with  $0 \leq i \leq d$ . To provide the desired security,  $d - 1$  shares are chosen uniformly at random while the remaining share is determined such that  $x = x_0 \circ x_1 \circ \dots \circ x_d$  holds. The group operator  $\circ$  is usually addition, either in  $\mathbb{F}_2$  (Boolean masking) or a larger field (additive masking). The parameter  $d$  defines the security order based on the  $d$ -probing model [13], where an attacker is assumed to obtain the exact values of up to  $d$  intermediate values of the target design. Hence, if the adversary does not learn anything about the secret values using  $d$  probes, the implementation is assumed to be secure against  $d$ -order attacks.

Functions that can be applied share-wise such that  $f(x) = f(x_0) \circ f(x_1) \circ \dots \circ f(x_d)$  are easy and efficient to mask. One of these linear functions is for example a XOR in the Boolean masking domain. Non-linear functions, for example, an AND, cannot be applied share-wise and need to be expressed differently. The challenge in masking cryptographic implementations relies upon avoiding or efficiently implementing non-linear functions.

## 2.3 Bitslicing

An important method for efficient Boolean-masked software implementations is bitslicing. Bitslicing changes the representation of values. Instead of storing  $n$  values in  $n$  distinct  $n$ -bit registers (32-bit in our case), we aggregate the  $i$ -th bit of each value in one register. This corresponds to a matrix transposition. If the maximum bit-length of the values is below the register width, bitslicing allows a condensed representation and simultaneously fewer Boolean instructions. Bitslicing is especially useful for algorithms that operate on single bits at



a time because it allows doing single-bit operations on  $n$  values simultaneously with one instruction, comparable to Single Instruction Multiple Data (SIMD) instructions. For masked implementations, bitslicing helps to reduce the number of costly non-linear operations.

## 2.4 Random Integer Sampling from Range

Sampling a uniform random integer from a given range is not always as simple as it seems. Both in software and hardware we can obtain uniform random bits from e.g., a Pseudorandom Number Generator (PRNG). By concatenating  $l$  random bits, we get a random value in the range of  $[0, 2^l)$ .

If we need a random value  $r$  in the range of  $[0, x)$  (which we denote with  $\text{rand}(x)$  in the following), where  $x$  is not a power of two, we can sample  $r$  from  $[0, 2^l)$ , with the smallest  $l$  such that  $x < 2^l$ , and reject  $r$  if it is not smaller than  $x$ . The closer  $x$  is to  $2^l$ , the fewer rejections occur, in the worst case, however, the chance for rejection is almost 50%.

Instead of rejecting values, one can alternatively use a function that maps the values from  $[0, 2^l)$  to  $[0, x)$ . An obvious function for this is computing  $r \bmod x$ . Given  $l$  random bits stored in  $r$  and a bit width of  $t$  for the target range  $x$  one can alternatively compute an  $(l + t)$ -bit multiplication  $rx$  and take the upper  $t$  bits of the result, which again will be a value between 0 and  $x - 1$ .

The drawback of both of these mapping methods is that they introduce a bias. When  $x$  is not a power of two,  $2^l$  will not divide  $x$ , therefore some values in the output range  $[0, x)$  will be more likely than others. With increasing  $2^l$  compared to  $x$ , the bias becomes neglectable and the output becomes close to uniform random.

If we want to sample an integer from a range  $[i, x)$  that is not starting at 0, we can use the previous methods and compute  $i + \text{rand}(x - i)$ .

## 2.5 Applications

Fixed weight polynomial sampling is a part of many PQC schemes, and many of them can potentially become (or already are) a standard determined by the National Institute of Standards and Technology (NIST).

*BIKE*. Bit Flipping Key Encapsulation (BIKE) has among three other KEMs advanced to the fourth round of NIST's standardization process and is a code-based scheme relying on Quasi-Cyclic Moderate-Density Parity-Check (QC-MDPC) codes. Polynomials live in the cyclic polynomial ring  $\mathcal{R} := \mathbb{F}_2[X]/(X^r - 1)$ , thus coefficients are either 0 or 1 and the number of coefficients is determined by the parameter  $r$  of the reduction polynomial. During key generation, two random fixed-weight polynomials are sampled:  $(h_0, h_1)$  with  $|h_0| = |h_1| = W/2$ . Moreover, during encapsulation and decapsulation, two fixed weight polynomials  $e_0, e_1$  are sampled with  $|e_0| + |e_1| = t$  where  $t$  is a publicly known and fixed parameter.

*HQC.* HQC also advanced to the fourth round of standardization. HQC also deploys fixed-weight sampling in key generation, encapsulation, and decapsulation. Analogously, polynomials in HQC have the polynomial ring  $\mathcal{R} := \mathbb{F}_2[X]/(X^r - 1)$ . Apart from parameters, the only difference then is that the polynomial  $e_0, e_1$  are sampled separately rather than with a joint fixed weight.

*McEliece.* The third remaining fourth-round candidate also uses fixed-weight sampling, but only during encapsulation to sample the “message”. McEliece is deemed to be the most conservative candidate during the whole standardization, being based on the more than 40 years old original McEliece cryptosystem.

*NTRU.* NTRU is a lattice-based Key Encapsulation Mechanism (KEM) and comes in two “flavors”: HRSS and HPS. For both, four polynomial rings are deployed. Fixed-weight sampling is used only during key generation of the HPS parameter sets. Furthermore, NTRU-HPS imposes the special requirement of having *exactly*  $W/2$  coefficients  $+1$  and  $W/2$  to be  $-1$ .

*Streamlined NTRU Prime and NTRU LPRime.* Streamlined NTRU Prime is a lattice-based KEM and is, together with X25519, currently the default algorithm for OpenSSH 9. NTRU LPRime is a merger with Streamlined NTRU Prime during the second round of NIST standardization. Both require fixed-weight sampling in their respective key generations, similar to NTRU with a ternary target space. However, no requirement is set on the number of  $+1$  and  $-1$ .

*Dilithium.* Dilithium is the designated PQC digital signature standard. It is based on the Module-Learning With Errors problem and operates on polynomials in the ring  $\mathbb{Z}_q[X]/(X^{256} + 1)$  with  $q = 8\,380\,417$ . Security is scaled through the matrix parameters. Being constructed with the help of the Fiat-Shamir with aborts technique, it simulates the verifier by querying a random oracle to sample a challenge during signature generation. This challenge has the specific form of a fixed-weight polynomial with ternary coefficients and no special restrictions on the number of coefficients with value  $-1$ . Based on several abort checks, a signature candidate may get rejected, starting over the whole signature generation including computing a new challenge  $c$ . Thus, it is not directly clear that  $c$  from rejected iterations is public information, even though the final  $c$  is part of the signature.

Previous work on the GLP signature scheme, which is a predecessor of Dilithium, has found that if the rejected challenges are viewed as public information together with their respective commitment, either one has to live with an additional heuristic security assumption or add a statistically hiding commitment scheme, tolerating the additional communication cost [3]. This is also stated regarding Dilithium in a recent preprint [1], where they state that rejected challenges are public and the commitment as well, but based on the Learning with Rounding assumption. To avoid this additional assumption, in our opinion it would be also feasible to perform the rounding masked, hashing  $w_1$  in masked domain, obtaining a masked bit-string  $\tilde{c}$ , which is already a representation of the challenge. This can then be unmasked (since we know that also rejected challenges are non-sensitive) and used to perform fixed-weight sampling.

### 3 Conceptual Considerations

Although the fixed weight polynomial sampling problem at its core is simple, its application comes with multiple problem dimensions depending on the algorithmic scheme, and implementation target.

*Attacker Model.* Sampling can be used in different parts of a KEM. If it is part of the key generation that is only executed once for one key, only single-trace side-channel attacks are applicable. The profiled Simple Power Analysis (SPA) is assumed as the strongest attacker model in our case.

Since in encapsulation, no secret key is used at all, usually no multi-trace attacks are eligible. In the current setting and applications, fixed weight sampling is used once during encapsulation to sample the message or an error. For decapsulation, multi-trace attacks are possible if the KEM key is non-ephemeral.

*Target Space.* Some use cases require binary polynomials while others sample ternary polynomials. For the ternary polynomials, it then can vary how the weight must be split between the ones and the minus ones.

*Target Representation.* The classic representation for a polynomial is an array of length  $N$  with one element for each coefficient (coefficient representation). However, polynomials can also be expressed by a list of non-zero indices (index representation). The cryptographic scheme may require different representations and the sampling methods output different representations. It is possible to convert one representation into the other.

*Determinism.* If the sampling is used in the encapsulation and decapsulation, it is usually required to provide the same output when given the same input seed. This can be achieved for all algorithmic approaches by using a PRNG as the source of randomness that is initialized with the seed. Determinism is usually not required in the key generation.

*Secret Seed.* In some use cases, the input seed for the PRNG is a secret value, thus the sampling algorithm must be constant-time not only with respect to the sampled polynomial but also with respect to the input seed. Concrete attacks have been presented recently in [12, 18].

*Parameters  $N$  and  $W$ .* The most important parameters that determine the performance of the sampling methods are the number of coefficients  $N$  and the number of non-zero coefficients  $W$  or the weight of the polynomial. In particular,  $N$  can vary distinctly from values between 256 to 81 194.

*Target Platform.* Implementing hardware or software influences the performance of an algorithm. Parallelism is important in either case, in software it can sometimes be achieved with bitslicing as introduced in Sect. 2.3, while in hardware, more fine-grained parallelism and trade-offs are possible.

### 3.1 Requirement Analysis

In Table 1 we give an overview of the most important parameters and requirements of each relevant scheme for the fixed-weight polynomial sampling.

The parameter sets of BIKE and HQC include relatively large  $N$  and small to medium  $W$ , and therefore a small  $W/N$  ratio which are all important factors for the sampling algorithms. Both schemes are also the only ones, that require

**Table 1.** Requirements for all potential applications

Scheme	Param.	Where?	$N$	$W$	$W/N$	Target Space	Det.	Sec. Seed
BIKE	L1	en/decaps	24646	134	0.005	binary	yes	yes
BIKE	L1	keygen	12323	71	0.006	binary	no	no
BIKE	L3	en/decaps	49318	199	0.004	binary	yes	yes
BIKE	L3	keygen	24659	103	0.004	binary	no	no
BIKE	L5	en/decaps	81194	264	0.003	binary	yes	yes
BIKE	L5	keygen	40973	137	0.003	binary	no	no
HQC	128	en/decaps	17669	75	0.004	binary	yes	yes
HQC	128	keygen	17669	66	0.004	binary	no	no
HQC	192	en/decaps	35851	114	0.003	binary	yes	yes
HQC	192	keygen	35851	100	0.003	binary	no	no
HQC	256	en/decaps	57637	149	0.003	binary	yes	yes
HQC	256	keygen	57637	131	0.003	binary	no	no
McEliece	348864	encaps	3488	64	0.018	binary	no	no
McEliece	460896	encaps	4608	96	0.021	binary	no	no
McEliece	6688128	encaps	6688	128	0.019	binary	no	no
McEliece	6960119	encaps	6960	119	0.017	binary	no	no
McEliece	8192128	encaps	8192	128	0.016	binary	no	no
NTRU	hps2048509	keygen	509	254	0.499	$W/2$ ternary	no	no
NTRU	hps2048677	keygen	677	254	0.375	$W/2$ ternary	no	no
NTRU	hps4096821	keygen	821	510	0.379	$W/2$ ternary	no	no
sNTRU Prime	653	keygen	653	288	0.441	uni. ternary	no	no
NTRU LPRime	653	keygen	653	252	0.386	uni. ternary	no	no
sNTRU Prime	761	keygen	761	286	0.376	uni. ternary	no	no
NTRU LPRime	761	keygen	761	250	0.329	uni. ternary	no	no
sNTRU Prime	857	keygen	857	322	0.376	uni. ternary	no	no
NTRU LPRime	857	keygen	857	329	0.384	uni. ternary	no	no
sNTRU Prime	953	keygen	953	396	0.416	uni. ternary	no	no
NTRU LPRime	953	keygen	953	345	0.362	uni. ternary	no	no
sNTRU Prime	1013	keygen	1013	448	0.442	uni. ternary	no	no
NTRU LPRime	1013	keygen	1013	392	0.387	uni. ternary	no	no
sNTRU Prime	1277	keygen	1277	492	0.385	uni. ternary	no	no
NTRU LPRime	1277	keygen	1277	429	0.336	uni. ternary	no	no

seed security and a deterministic sampling algorithm for their encapsulation and decapsulation. Their polynomials have coefficients that are either 0 or 1, this is also the case for McEliece. NTRU on the other hand has ternary coefficients that are either 0, 1 or  $-1$  and the fixed number of nonzero coefficients  $W$  must be equally split between the 1s and  $-1$ s. For the ternary coefficients in Streamlined NTRU Prime and NTRU LPRime, this relation is uniformly random. The schemes with ternary coefficients also have in common that the sampling is only used during the key generation, security against single-trace side-channel attacks is therefore sufficient.

## 4 Designing Masked Fixed Weight Sampling

In the following sections, we present multiple side-channel secure approaches for fixed-weight polynomial sampling. The different approaches can be categorized into three different groups. The rejection method in Sect. 4.4 and its bounded variant in Sect. 4.5 solve the problem by sampling  $W$  distinct values in the range  $[0, N)$ , which represent the indices of the non-zero coefficients. The methods based on Fisher-Yates in Sect. 4.2 and sorting in Sect. 4.3 utilize shuffling of fixed input polynomials. ANDING in Sect. 4.7 and our comparison method in Sect. 4.6 both sample fixed-weight polynomials by randomly setting bits.

For each approach, we start by explaining the fundamental idea, then we clarify how to achieve a timing side-channel secure (constant-time) variant that is a necessity for a power side-channel secure implementation. Based on this, we explain how to realize a masked and efficient variant. In Sect. 5, we provide more details about our implementations. We present the algorithms only for the binary use case, in most cases they can easily be adapted for the ternary use case. If this adoption is not obvious, we explain how it can be achieved. Some of the algorithms have a small bias, so their output is only close to uniform random. Before actually deploying a scheme with one of the biased methods one needs to diligently prove that the bias does not impair the security.

**Masked Sampling by Coron et al.** In recent work [9], Coron et al. present an approach of side-channel-secure fixed-weight sampling for NTRU, which proposes the following strategy:

1. Initialize an empty polynomial with the first  $W/2$  coefficients set to  $-1$ , the subsequent  $W/2$  coefficients to  $+1$ , and the remaining coefficients set to 0.
2. Generate a fresh arithmetic masking of this polynomial.
3. Shuffle each share with the same permutation.
4. Re-share the arithmetic sharing.
5. Repeat the last two steps a total of  $d + 1$  times, every time using a new permutation.

This high-level procedure is proven to be secure in the  $d$ -probing model. For their proof, however, the applied permutation is assumed to be a black box. Thus, we believe that it will be very hard, if not impossible, to instantiate securely

in practice. Moreover, Karabulut *et al.* show a single-trace attack that targets the permutation itself [14] and there is no reason to believe that an attacker is not able to attack multiple subsequent executions of different permutations successfully. Hence, it is reasonable to assume that this countermeasure does not protect against SPA attackers comprehensively.

#### 4.1 Core Operations

The masked algorithmic approaches for fixed-weight polynomial sampling that we present in the following sections, share a small set of operations that are repeatedly used and contribute distinctly to the overall performance. In this section, we explain how to perform a masked conditional move and different integer comparisons in the Boolean domain with little non-linear operations.

**Conditional Move in Boolean Domain.** A very important building block for our masked algorithms is the conditional move. The semantic of  $\text{cmov}(d, s, c)$  is that  $d$  is overwritten by  $s$ , if the condition flag  $c$  is set and  $d$  remains unchanged if  $c$  is 0.

For non-masked, but constant-time implementations, a conditional move is most efficiently expressed in software with a dedicated instruction, but can generally be expressed with a short sequence of arithmetic or Boolean instructions to avoid branching on the secret condition  $c$  and thus leak  $c$  via timing differences. A straightforward sequence would be  $d = (d \wedge \neg c) \vee (s \wedge c)$ .

This solution is, however, costly to mask, because it includes three non-linear operations, two ANDs, and one OR. It is possible to reduce the number of non-linear operations to one by using XOR operations:  $d = d \oplus ((d \oplus s) \wedge c)$  evaluates to  $d = d \oplus d \oplus s = s$ , if  $c$  is true and to  $d = d \oplus 0 = d$ , if  $c$  is false.

**Integer Comparison in Boolean Domain.** Let  $a[l-1:0], b[l-1:0]$  be bit vector variables representing integers in the range  $[0, 2^l)$ . To check whether  $a < b$ , we can simply compute  $a - b$  and then check whether the result is negative, in which case we know that  $a < b$ , and else,  $a \geq b$ . Thus, in Boolean domain, we can employ a Ripple-Carry subtractor which computes  $r[l:0] = a[l-1:0] - b[l-1:0]$ . Then,  $r[l]$  is the uppermost carry-out bit, which decides whether or not the result is negative. The Ripple-Carry subtractor performs the following computations:

$$r[0] = a[0] \oplus b[0] \tag{2}$$

$$c[0] = \overline{a[0]} \wedge b[0] \tag{3}$$

$$r[i] = a[i] \oplus b[i] \oplus c[i-1] \quad \forall 1 \leq i < l \tag{4}$$

$$c[i] = (c[i-1] \wedge (\overline{a[i]} \oplus b[i])) \oplus (\overline{a[i]} \wedge b[i]) \quad \forall 1 \leq i < l \tag{5}$$

$$r[l] = c[l-1] \tag{6}$$

This is usually done in Central Processing Units (CPUs), where the subtraction instruction is also used for integer comparison, but without writing the

result back to the registers. In the masked case, however, we aim to achieve a very low number of secure AND gates. Thus, as we only want to recover  $r[l]$  rather than the full subtraction result, we propose an alternative approach.

$t = a \oplus b$  gives us the bits, where  $a$  and  $b$  differ. The highest set bit of  $t$  determines the bit or rather the index  $g$  in  $a$  and  $b$  that determines which of the two variables is greater. Because we know that  $a$  and  $b$  differ at this bit, it is enough to look at one of them. E.g. if  $b_g$  is set,  $b$  is greater than  $a$ . To perform this concept in constant-time we iterate over all bits, starting from the lowest bit, and update our output with  $b_i$  if  $t_i$  is set, which ultimately results in  $b_g$  in our output. With our output initialized with 0, it will result in 0 if  $a \geq b$  and 1 if  $a < b$ . Algorithm 1 describes this idea formally.

At first sight, Algorithm 1 does not need any expensive AND gadgets, but for implementing the conditional move securely we require one AND, as explained in the previous subsection. Compared to the traditional approach via subtraction, we can half the amount of expensive non-linear gadgets. The overall asymptotic runtime is determined by the bit length of the inputs. In the algorithms presented in the following, we compare values bounded by  $N$ , so the cost for one comparison is  $\lceil \log_2(N) \rceil$ .

---

**Algorithm 1.** Optimized Integer Comparison in Boolean Domain

---

**Require:**  $a = \sum_{i=0}^l a_i 2^i$ ,  $b = \sum_{i=0}^l b_i 2^i$

**Ensure:**  $res \leftarrow a < b ? 1 : 0$

**function** CMPL( $res, a, b$ )

$t \leftarrow a \oplus b$

$res \leftarrow 0$

**for**  $i \leftarrow 0$  **to**  $l$  **do**

        cmov( $res, b_i, t_i$ )

$\triangleright res := res \oplus ((res \oplus b_i) \wedge t_i)$

**end for**

**end function**

---

**Comparison with Fixed Public Input.** We can simplify this further when we have one fixed and public input  $b$  rather than two variable ones. Then, to compare whether or not  $a < b$ , we first employ the same procedure as in Algorithm 1. However, for each  $t_i$ , we now know publicly that it is either

- $a_i$  in the case that  $b_i = 0$ , or
- $\neg a_i$  in the case that  $b_i = 1$ .

Thus, we have

- for  $b_i = 0$ ,  $res := res \oplus ((res \oplus 0) \wedge a_i) = res \wedge \neg a_i$ , and
- for  $b_i = 1$ ,  $res := res \oplus ((res \oplus 1) \wedge \neg a_i) = res \vee \neg a_i$ .

This does not save non-linear gates, as we still need one per bit, but it saves several XOR operations, which are cheap, but not free. Moreover, we can completely omit all lower bits until the first  $b_i = 1$ , since we start with  $res = 0$ , which sets all subsequent intermediate  $res$  to zero.

**Comparison on Equality.** Evaluating whether two masked values are equal or not is even cheaper to realize in the Boolean domain.  $c = a \oplus b$  is only zero if  $a$  is equal to  $b$ . Thus we can iterate over all bits in  $c$  and condense them to one bit with masked OR operations. After flipping the resulting bit,  $res$  will be one, if  $c$  is zero and thus  $a$  is equal to  $b$  and zero otherwise, denoted with  $cmpeq(res, a, b)$  in the following. The asymptotic runtime cost is again  $\mathcal{O}(\log_2(N))$ .

## 4.2 Fisher-Yates

The Fisher-Yates shuffle is an algorithm to get a uniform random permutation of a fixed input sequence in  $\mathcal{O}(N)$  time. Similar to the sorting approach explained in Sect. 4.3 it can be directly applied to a fixed polynomial with the correct weight to get a random polynomial with the correct weight.

Alternatively, one can apply Fisher-Yates to an array with length  $N$  with distinct integers from 0 to  $N$  and treat the first  $W$  elements of the output as the indices respectively the coefficients of the polynomials which are non-zero. In this case, the permutation of the elements beyond the first  $W$  elements is irrelevant and the algorithm can be stopped after  $W$  iterations because the first  $W$  elements are not affected by further shuffling.

In its original version, Fisher-Yates is not timing side-channel-secure, because its memory accesses reveal the permutation and (only relevant for a secret seed) it requires uniform random numbers from a varying range, which requires a rejection step.

Sendrier [18] tackled these problems with two modifications. First of all, he showed for BIKE that the security of the cryptographic scheme is not necessarily impaired when the sampling is only close to uniform random if the parameters are correctly chosen. This eliminates the need for the rejection step by allowing a slightly biased constant-time approach as explained in Sect. 2.4. The secret dependent memory accesses can also be circumvented, but this comes with quadratic runtime instead of the original linear runtime. The solution for the index sampling method is depicted in Algorithm 2.

---

### Algorithm 2. Constant-Time Fisher-Yates [18]

---

**Require:**  $N, W$

**Ensure:**  $W$  distinct elements of  $0, \dots, N - 1$

```

function FISHER-YATES( $N, W$ )
  for  $i \leftarrow 0$  to  $W - 1$  do
     $p[i] \leftarrow i + \text{rand}(N - i)$ 
  end for
  for  $i \leftarrow W - 1$  to  $0$  do
    for  $j \leftarrow i + 1$  to  $W - 1$  do
       $cmpeq(cond, p[j], p[i])$ 
       $cmov(p[j], i, cond)$ 
    end for
  end for
end function

```

---



For masking the constant-time Fisher-Yates algorithm two components need to be protected. The first component is sampling a random integer in the range of  $[0, N - i)$ . Sendrier [18] proposed to compute a random value  $r \bmod N - i$ , but the implied division is a costly operation. Furthermore, in most CPUs, a division is an instruction with a variable cycle count depending on the input and thus not constant time. A modulo reduction with a constant modulo might be translated by a compiler to a constant-time Barrett reduction, but there is no guarantee for this.

We propose to use the faster multiplication approach as explained in Sect. 2.4 instead. Multiplication instructions are constant-time for most CPUs. In the additive masking domain, the multiplication with the public range value and the addition of the public index  $i$  can be efficiently performed sharewise.

The second component is the comparison of equality and the following conditional move, both can be done in Boolean domain, therefore a transformation from arithmetic to Boolean domain between the two components is necessary. The inner loop in Algorithm 2 containing the comparison and conditional move can be computed in parallel for multiple  $j$ , because the iterations are independent of each other.

A masked implementation of this Fisher-Yates algorithm results in an asymptotic runtime of  $\mathcal{O}(W^2 \log_2(N))$ , for sampling a close-to-uniform polynomial in the index representation without leaking a secret seed.

### 4.3 Sorting

An alternative approach to obtain a uniformly random permutation of a set is to attach distinct random values to each element and sort the pairs according to the random value. Bernstein [5] suggested applying this principle to sampling fixed-weight polynomials by starting with a polynomial with the desired weight and then getting a random permutation by sorting.

---

#### Algorithm 3. Sort based Sampling [5]

---

**Require:**  $N, W, l, p[N]$

**Ensure:** random bitpolynomial in  $p[N]$  with weight  $W$

**function** SORTSAMPLING( $N, W$ )

**for**  $i \leftarrow 0$  **to**  $N - 1$  **do**

**if**  $i < W$  **then**

$t \leftarrow 1$

**else**

$t \leftarrow 0$

**end if**

$r \leftarrow \text{rand}(2^l)$

$p[i] \leftarrow (r \lll 1) + t$

**end for**

$\text{sort}(p)$

**for**  $i \leftarrow 0$  **to**  $N - 1$  **do**

$p[i] \leftarrow p[i] \wedge 1$

**end for**

**end function**

---

To get *distinct* random values one can use rejection sampling, e.g., for each new randomly sampled value one checks if it collides with one of the values sampled before. If yes, the new value gets rejected and one continues until enough distinct values are sampled. Bernstein showed that the rejection step can be skipped if the size of the random value is big enough compared to the number of elements such that the chance of a collision becomes neglectable. With a constant-time sorting algorithm, the entire procedure is constant-time with respect to the sampled polynomial and the seed for the PRNG. The runtime depends on the implementation of the sorting algorithm and the polynomial size  $N$  as a parameter. This approach can be directly applied to sampling binary and ternary polynomials.

Sorting algorithms can have at lowest linear asymptotic runtime, but then usually no efficient constant-time implementation exists. A group of sorting algorithms that can be very efficiently implemented in constant-time is sorting networks, they consist only of a fixed number of comparisons and swaps. Comparison-based sorting algorithms have at best an asymptotic runtime of  $\mathcal{O}(N \log(N))$ . A naive masked implementation of a sorting network mainly consists of a comparison and a conditional move depending on the comparison, both can be masked efficiently in software and in hardware.

The sorting approach is deployed in NTRU and Streamlined NTRU Prime [6] with an implementation based on Batcher’s Odd-Even mergesort [4]. For our implementation, we opted for Batcher’s Bitonic mergesort [4], because it is easier to parallelize in the bitsliced domain, which is critical for our efficient masked software implementation. Both sorting algorithms have an asymptotic runtime of  $\mathcal{O}(N \log^2(N))$ . Although we use our improved comparison approach explained in Sect. 4.1 instead of a costly subtraction, the masked comparison and the conditional move are still the overwhelming driver in cycle costs.

A major drawback of this sampling method besides its high runtime costs for large polynomial size  $N$  is the high amount of randomness required upfront resulting also in high memory usage, compared to other methods. This can be circumvented by using radixsort. Radixsort utilizes an arbitrary, stable sorting algorithm to sort numbers e.g., bit by bit, starting from the lowest bit. The stableness of the sorting algorithm ensures that the order according to the lower bits is maintained when sorting according to the higher bits<sup>2</sup>. As radixsort only works on one bit per sorting iteration, only one random bit per element needs to be sampled and stored at a time because we are not interested in the sorted random values, but only in the permutation the sorting provides. Stable sorting networks exist, but they have a quadratic asymptotic runtime, which makes this approach more costly. Radixsort combined with an unstable sorting network does not result in correct sorting universally and coherently also not in uniform random permutations, which we confirmed for small parameters by exhaustive testing.

---

<sup>2</sup> Stable sorting in ascending manner according to the MSB of (10, 11, 01) results in (01, 10, 11) and not (01, 11, 10).

#### 4.4 Rejection Sampling

Probably the most obvious solution for fixed-weight polynomial sampling is the rejection method. One samples a uniform random value  $r$  below  $N$  by rejecting values from the range  $[0, 2^l)$ , with the smallest  $l$  such that  $x < 2^l$ . Then one iterates over the already sampled indices and checks for a collision, if a collision is found,  $r$  gets rejected. The rejection sampling continues until  $W$  distinct indices are sampled as presented in Algorithm 4.

The runtime of this probabilistic algorithm varies and depends on the randomness, therefore it is not suitable for cryptographic schemes, where the seed for the PRNG is secret. This restriction in application allows early termination of loops, as soon as the rejection becomes evident. Although the result of the comparisons for equality for the collision check is public, we cannot XOR both arguments and then simply unmask the result and check if it is zero or not. In this case, we would leak the bits in which  $r$  differs from  $p[c]$ . So the comparisons themselves must be side-channel secure to protect the non-rejected values. This can be done with the core operations presented in Sect. 4.1.

---

#### Algorithm 4. Rejection Sampling - Index

---

**Require:**  $N, W, 2^l > N, p[W]$

**Ensure:**  $W$  distinct elements of  $0, \dots, N - 1$

```

function REJECTION-INDEX( $N, W$ )
   $i \leftarrow 0$ 
  while  $i < W$  do
     $r \leftarrow \text{rand}(2^l)$ 
     $\text{cml}(t, r, N)$ 
    if  $\neg t$  then
      continue
    end if
     $\text{collision} \leftarrow 0$ 
    for  $c \leftarrow 0$  to  $i - 1$  do
       $\text{cmpeq}(t, p[c], r)$ 
      if  $t$  then
         $\text{collision} \leftarrow 1$  break
      end if
    end for
    if  $\text{collision} = 1$  then
      continue
    end if
     $p[i] \leftarrow r$ 
     $i \leftarrow i + 1$ 
  end while
end function

```

---

For this algorithm,  $N$  determines the probability for the first rejection step, with an  $N$  only slightly greater than the closest power of two this probability

can be close to 50%.  $W/N$  determines the probability of the second rejection when checking for collisions. With a  $W/N$  close to 0.5, the chance for a collision for a single value gets close to 50% for the last iterations when  $i$  reaches  $W$ , so on average the probability for rejection due to a collision for a single value can be up to 25%. Drucker et al. [10] already pointed out that the fixed weight polynomial sampling problem is symmetric such that for  $W/N > 0.5$ , one can solve it for  $(N - W)/N$  and invert the result.

#### 4.5 Bounded Rejection Sampling

The idea of a bounded rejection sampling algorithm as presented by Drucker et al. [15] for the BIKE use case, is to transform the rejection sampling method as presented in Sect. 4.4 such that it is constant-time also with respect to the PRNG seed. This idea has also been implemented similarly by Guo et al. [12] for HQC.

For this, the rejections must not influence the path taken in the algorithm and therefore branches in a software implementation and the memory access pattern must be independent of the randomness. This is done by keeping track of the number of valid samples with a secret counter that indicates where to input the next valid index into the array and does not get incremented if a sample gets rejected so that the next sample can overwrite the rejected one. Early termination of loops is not possible anymore, so with every sampled value one has to iterate over the entire array of indices and securely check for a collision. These comparisons can however be performed in parallel. Also, the comparison of the current index with the counter and the conditional move can be parallelized, as the comparison only outputs 1 for a single index for one complete iteration

---

#### Algorithm 5. Bounded Rejection Sampling - Index [15]

---

**Require:**  $N, W, B, 2^l > N, p[W]$

**Ensure:**  $W$  distinct elements of  $0, \dots, N - 1$

**function** BOUND-REJECTION-INDEX( $N, W, B$ )

$cntr \leftarrow 0$

**for**  $i \leftarrow 0$  **to**  $B - 1$  **do**

$r \leftarrow \text{rand}(2^l)$

$dup \leftarrow 0$

**for**  $c \leftarrow 0$  **to**  $W - 1$  **do**

$\text{cmpeq}(t, p[c], r)$

$dup \leftarrow dup \vee t$

$\text{cmpeq}(f, c, cntr)$

$\text{cmov}(p[c], r, f)$

**end for**

$\text{cmpl}(t, r, N)$

$t \leftarrow !dup \wedge t$

$cntr \leftarrow cntr + t$

**end for**

**end function**

---

over the array. Thus the counter can be conditionally incremented only once after iterating over the array and remains constant during the loop.

The second challenge for any seed-independent timing is the number of random values that need to be sampled which can not be determined exactly upfront, but they can be estimated. Depending on the parameters  $N$  and  $W$  one can compute a loose upper bound  $B$  of iterations or rather samples, within with overwhelming probability at least  $W$  valid indices are found.

The majority of the algorithm can be masked with Boolean components that we already discussed in previous algorithms. The incrementation of the secret counter is most efficient in the additive masking domain, however, the counter is also required in the Boolean domain for the comparison. To avoid the costly transformations between the domains, we propose performing the addition with a single bit in the Boolean domain with half-adders implying  $\lceil \log_2(N) \rceil$  masked ANDs.

Algorithm 5 demonstrates this approach, since  $B$  is a multiple of  $W$  the runtime is  $\mathcal{O}(W^2 \log_2(N))$ . The asymptotic view indicates a similar performance to the Fisher-Yates algorithm, but a closer inspection reveals that first, bounded rejection takes more than  $W^2$  iterations compared to  $\frac{1}{2}W^2$  for Fisher-Yates and the rejection method requires two masked comparisons for each iteration compared to one comparison for Fisher-Yates. When the sampling  $\text{rand}(N - i)$  of  $W$  values in Fisher-Yates does not contribute significant costs, the bounded rejection is probably less performant when masked.

---

**Algorithm 6.** Bounded Rejection Sampling - Coefficient

---

**Require:**  $N, W, B, 2^l > N, p[N]$  initialized with zeros

**Ensure:**  $W$  random coefficients in  $p$  are set to 1

**function** BOUND-REJECTION-COEFF( $N, W, B$ )

$cntr \leftarrow 0$

**for**  $i \leftarrow 0$  **to**  $B - 1$  **do**

$t \leftarrow 0$

$r \leftarrow \text{rand}(2^l)$

$\text{cmpeq}(f0, cntr, W)$

**for**  $c \leftarrow 0$  **to**  $N - 1$  **do**

$\text{cmpeq}(f1, c, r)$

$\text{cmpeq}(f2, p[c], 0)$

$f \leftarrow \neg f0 \wedge f1 \wedge f2$

$\text{cmov}(p[c], 1, f)$

$t \leftarrow t \vee f$

**end for**

$cntr \leftarrow cntr + t$

**end for**

**end function**

---

Alternatively, the sampling of values less than  $N$  can be realized with the biased multiplication method as we use it for Fisher-Yates. For some parameter

sets of HQC, this might be faster, because  $N$  is close to the next lower power of two, thus the chance of rejection when  $r \geq N$  is high and the upper bound  $B$  is higher. In this case, however, the runtime comparison to Fisher-Yates is even more clear and indicates that Fisher-Yates is the faster solution.

In Algorithm 6 we show how the bounded rejection method can be adapted to output polynomials in the coefficient representation instead of the index representation with asymptotic runtime  $\mathcal{O}(WN \log_2 N)$ .

The bounded rejection sampling is only relevant for cryptographic schemes, where the PRNG input needs to be protected as the protection comes with a performance overhead compared to the simple rejection method.

## 4.6 Comparison Sampling

The idea of this novel approach is to sample each coefficient of the polynomial individually with an approximation of the probability  $W/N$ . This can be implemented efficiently by comparing a uniform random bit string of length  $\ell$  with a fixed threshold  $t$  such that  $t/2^\ell \approx W/N$ . If  $t$  is smaller than the random  $\ell$ -bit value, the coefficient is set to 1.

After performing this for each coefficient, a masked weight check of the polynomial is carried out and the polynomial is accepted only if the correct weight  $W$  is hit. Else, the whole procedure is repeated, which renders this approach infeasible for use cases that require runtime independent of the input seed. This method can be considered somewhat of a generalization of the RepeatedAND method by Drucker and Gueron that we cover in Sect. 4.7.

**Table 2.** BIKE Comparison sampling, for number of expected repetitions and expected random bits, see Eqs. 7 and 8

$\ell$	BIKE-L1			BIKE-L3			BIKE-L5		
	$t$	rep.	rnd.	$t$	rep.	rnd.	$t$	rep.	rnd.
8	1	2439.74	240 518 881	1	31.88	6 289 754	1	169.06	55 415 054
9	3	21.30	2 362 385	2	31.88	7 075 973	2	169.06	62 341 935
10	6	21.30	2 624 872	4	31.88	7 862 192	3	92.48	37 892 643
11	12	21.30	2 887 359	9	29.10	7 892 628	7	30.30	13 658 519
12	24	21.30	3 149 847	17	25.46	7 533 925	14	30.30	14 900 203
13	47	21.10	3 379 948	34	25.46	8 161 752	27	29.73	15 833 552
14	94	21.10	3 639 944	68	25.46	8 789 579	55	29.34	16 829 906

For efficiency, the choice of  $\ell, t$  is decisive and the expected number of repetitions of the overall procedure is determined by

$$\mathcal{B}(W, N, t/2^\ell)^{-1} \tag{7}$$

Therefore, these parameters must be chosen carefully for each potential use case.

Let  $p = W/N$  be the target probability. Then, for  $\ell$  random bits, the best comparison threshold  $t$  is  $\lfloor p2^\ell \rfloor$ . Intuitively, the larger we choose  $\ell$ , the better we approximate  $p$  at cost of more randomness and more secure operations. Interestingly, for all applications, there exists a threshold for  $\ell$ , from which increasing does not improve the success probability significantly.

Apart from minimizing the number of non-linear operations, we also want to minimize the number of fresh random bits that are required. For a given  $(N, W, \ell, t)$ , we know that

$$\mathcal{B}(W, N, t/2^\ell)^{-1} \cdot N\ell \tag{8}$$

is the expected number of fresh random bits for this method, which will help us choose  $\ell, t$  for each use case. On the lower layer, we can employ our efficient comparison from Algorithm 1 and the optimizations for comparison with one fixed operand, resulting in  $\ell - 1$  non-linear operations per coefficient and  $\mathcal{B}(W, N, t/2^\ell)^{-1} \cdot N(\ell - 1)$  expected non-linear operations overall for a given  $(N, W, \ell, t)$ .

Note that these numbers refer to the *unprotected* instantiation. When masking this approach, we require  $d + 1$  times as much randomness and in addition, fresh randomness for each non-linear operation.

In the following, we give details on each potential application.

**BIKE and HQC Key Generation.** For BIKE and HQC, we cannot deploy this method for encapsulation and decapsulation, due to the attack by [12, 18]. Still, it is eligible for key generation in both cases. Table 2 and Table 3 give details on the choice of  $\ell, t$  for both algorithms. As can be seen there, for BIKE-L1  $\ell = 9, t = 3$  is the obvious choice, as well as  $\ell = 8, t = 1$  for BIKE-L3 and  $\ell = 11, t = 7$  for BIKE-L5.

For HQC,  $\ell = 8, t = 1$  is the best choice for HQC-128,  $\ell = 10, t = 3$  for HQC-196, and  $\ell = 12, t = 9$  for HQC-256. Moreover, the randomness numbers indicate that BIKE performs better with this approach.

**Table 3.** HQC Comparison Sampling, for number of expected repetitions and expected random bits, see Eqs. 7 and 8

$\ell$	HQC-128			HQC-196			HQC-256		
	$t$	rep.	rnd.	$t$	rep.	rnd.	$t$	rep.	rnd.
8	1	21.77	3 076 984	1	1.5e4	4 270 634 825	1	3.7e11	1.7e17
9	2	21.77	3 461 607	1	7272.20	2 346 440 182	1	120.43	62 473 484
10	4	21.77	3 846 230	3	28.33	10 155 736	2	120.43	69 414 983
11	8	21.77	4 230 853	6	28.33	11 171 310	5	40.46	25 649 983
12	15	20.62	4 371 146	11	26.90	11 572 734	9	30.89	21 361 556
13	31	20.47	4 700 992	23	25.11	11 700 990	19	29.46	22 076 057
14	61	20.36	5 036 069	46	25.11	12 601 066	37	28.75	23 201 162

**Table 4.** McEliece Comparison Sampling, for number of expected repetitions and expected random bits, see Eqs. 7 and 8

$\ell$	Parameter Set														
	348864			460896			6688128			6960119			8192128		
	$t$	rep.	rnd.	$t$	rep.	rnd.	$t$	rep.	rnd.	$t$	rep.	rnd.	$t$	rep.	rnd.
6	1	44.13	923655	1	965.40	26691249	1	344.42	13820908	1	43.68	1823905	1	28.16	1383882
7	2	44.13	1077597	3	49.42	1593954	2	344.42	16124392	2	43.68	2127889	2	28.16	1614530
8	5	22.66	632174	5	29.70	1094851	5	28.88	1544999	4	43.68	2431873	4	28.16	1845177
9	9	21.11	662545	11	25.49	1057213	10	28.88	1738124	9	28.43	1780973	8	28.16	2075824
10	19	19.98	696744	21	24.62	1134475	20	28.88	1931249	18	28.43	1978859	16	28.16	2306471

**Table 5.** NTRU HPS Comparison Sampling, for number of expected repetitions and expected random bits, see Eqs. 7 and 8. Note that these are the numbers for generating a masked *binary* polynomial. In Sect. 4.6 we explain the transformation into a ternary.

$\ell$	Parameter Set								
	2048509			2048677			4096821		
	$t$	rep.	rnd.	$t$	rep.	rnd.	$t$	rep.	rnd.
1	1	28.32	14 414	1	5.73e+10	3.878e+13	1	1.32e+12	1.086e+15
2	2	28.32	28 827	2	5.73e+10	7.757e+13	2	1.32e+12	2.172e+15
3	4	28.32	43 241	3	31.59	64 164	5	35.75	88 043
4	8	28.32	57 655	6	31.59	85 551	10	35.75	11 7391
5	16	28.32	72 069	12	31.59	106 939	20	35.75	146 739
6	32	28.32	86 482	24	31.59	128 327	40	35.75	176 087
7	64	28.32	100 896	48	31.59	149 715	80	35.75	205 435
8	128	28.32	115 310	96	31.59	171 103	159	34.85	228 911

**Table 6.** Streamlined NTRU Prime Comparison Sampling, for number of expected repetitions and expected random bits, see Eqs. 7 and 8

$\ell$	Parameter Set											
	653		761		857		953		1013		1277	
	$t$	rnd.	$t$	rnd.	$t$	rnd.	$t$	rnd.	$t$	rnd.	$t$	rnd.
1	1	1966846	1	5.1e+14	1	1.3e+16	1	3.1e+10	1	3.5e+07	1	3.0e+19
2	2	3933692	2	1.0e+15	2	2.5e+16	2	6.3e+10	2	7.0e+07	2	6.0e+19
3	4	5900538	3	76571	3	91488	3	2945795	4	1.1e+08	3	222512
4	7	84497	6	102095	6	121985	7	371651	7	168223	6	296683
5	14	105621	12	127618	12	152481	13	215382	14	210278	12	370853
6	28	126745	24	153142	24	182977	27	235987	28	252334	25	360750
7	56	147869	48	178666	48	213473	53	255543	57	286495	49	396206

*BIKE-L1 Optimization.* We have  $\ell = 9, t = 3$  and thus want to have  $a = 2^9 - 4$  in Algorithm 1 to obtain a 1 output bit in  $3/2^9$  cases for a random input  $b$ . Then, we apply the above-described optimizations for a fixed input comparison:



**Table 7.** NTRU LPRime Comparison Sampling, for number of expected repetitions and expected random bits, see Eqs. 7 and 8

$\ell$	Parameter Set											
	653		761		857		953		1013		1277	
	$t$	rnd.	$t$	rnd.	$t$	rnd.	$t$	rnd.	$t$	rnd.	$t$	rnd.
1	1	5.7e+11	1	1.7e+24	1	4.1e+14	1	3.3e+20	1	8.6e+15	1	1.4e+35
2	2	1.1e+12	1	6.4e+09	2	8.2e+14	1	4.1e+17	2	1.7e+16	1	1.8e+15
3	3	72090	3	2643045	3	106026	3	150085	3	160778	3	11021700
4	6	96120	5	155129	6	141368	6	200113	6	214370	5	1083789
5	12	120150	11	183384	12	176711	12	250142	12	267963	11	321266
6	25	126010	21	148387	25	199178	23	215790	25	243060	22	385519
7	49	144495	42	173118	49	214613	46	251755	50	283570	43	378276
8	99	163109	84	197849	98	245272	93	284544	99	315026	86	432315

$$\begin{aligned}
r &= ((0 \vee b_0) \vee b_1) \wedge b_2 \wedge b_3 \wedge b_4 \wedge b_5 \wedge b_6 \wedge b_7 \wedge b_8 \\
&= (b_0 \vee b_1) \wedge \bigwedge_{i=2}^8 b_i = \neg(\neg b_0 \wedge \neg b_1) \wedge \bigwedge_{i=2}^8 b_i
\end{aligned}$$

Note that we convert the logical OR into a logical AND by De Morgan's law since this is how it is implemented with masked gadgets. Inversion is  $\mathcal{O}(1)$ , while  $\text{SecAnd}$  is  $\mathcal{O}(d^2)$ , so this does not increase asymptotic complexity. Still, we can save two inversions, since  $b_0, b_1$  are random input bits, which we can assume to be inverted already. It follows that for BIKE-L1, the following Boolean formula can be used to obtain a random bit with approximately the correct probability of being one, using random input bits  $b_0, \dots, b_8$ .

$$r = \neg(b_0 \wedge b_1) \wedge \bigwedge_{i=2}^8 b_i \quad (9)$$

*BIKE-L3 and HQC-128 Optimization.* With  $\ell = 8, t = 1$ , we fall back to the repeated AND method and can just compute  $r = \bigwedge_{i=0}^7 b_i$  for uniform random bits  $b_0, \dots, b_7$ . Notably, we can use this approach both for BIKE-L3 and HQC-128.

*BIKE-L5 Optimization.* With  $\ell = 11, t = 7$ , we have  $a = 2^{11} - 8$  in Algorithm 1 with random bits  $b_0, \dots, b_{10}$ . Then, applying the analog optimizations as above, including not inverting random input bits:

$$\begin{aligned}
r &= \left( \bigvee_{i=0}^2 b_i \right) \wedge \bigwedge_{i=3}^{10} b_i = \neg \left( \bigwedge_{i=0}^2 \neg b_i \right) \wedge \bigwedge_{i=3}^{10} b_i \\
&\sim \neg \left( \bigwedge_{i=0}^2 b_i \right) \wedge \bigwedge_{i=3}^{10} b_i
\end{aligned} \quad (10)$$

*HQC-196 Optimization.* Using  $\ell = 10, t = 3$ , we set  $a = 2^{10} - 4$  in Algorithm 1 with random bits  $b_0, \dots, b_9$ . Applying the aforementioned optimizations, we obtain

$$\begin{aligned} r &= (b_0 \vee b_1) \wedge \bigwedge_{i=2}^9 b_i = \neg(\neg b_0 \wedge \neg b_1) \wedge \bigwedge_{i=2}^9 b_i \\ &\sim \neg(b_0 \wedge b_1) \wedge \bigwedge_{i=2}^9 b_i \end{aligned} \quad (11)$$

*HQC-256 Optimization.*  $\ell = 12, t = 9$  implies setting  $a = 2^{12} - 10$  in Algorithm 1 with random bits  $b_0, \dots, b_{11}$ .

$$\begin{aligned} r &= \left( \left( \bigwedge_{i=0}^2 b_i \right) \vee b_3 \right) \wedge \bigwedge_{i=4}^{11} b_i = \neg \left( \neg \left( \bigwedge_{i=0}^2 b_i \right) \wedge \neg b_3 \right) \wedge \bigwedge_{i=4}^{11} b_i \\ &\sim \neg \left( \neg \left( \bigwedge_{i=0}^2 b_i \right) \wedge b_3 \right) \wedge \bigwedge_{i=4}^{11} b_i \end{aligned} \quad (12)$$

**McEliece Encapsulation.** For this application, we have no special restrictions, which renders the Comparison approach possible. As can be seen from Table 4, there are feasible choices of  $\ell, t$  for each parameter set. Notably, the highest parameter set has both  $N$  and  $W$  set to a power of two, which implies that the Comparison method falls back effectively to the RepeatedAND method.

**NTRU, Streamlined NTRU Prime, and NTRU LPRime Key Generation.**

For NTRU, Streamlined NTRU Prime and NTRU LPRime, we have an interesting different case, since the target space is not binary, but ternary. Additionally, NTRU imposes the condition that *exactly*  $W/2$  coefficients need to be  $+1$  and the remaining  $-1$ . To convert a binary polynomial to a ternary one, we employ the following strategy, assuming that we already have sampled a Boolean masked, weight- $W$  polynomial:

1. Sample a uniform random, masked bit  $r_i$  for each coefficient  $a_i$  with  $0 \leq i < N$ .
2. Compute securely the masked sign  $s_i := r_i \wedge a_i$  for each masked coefficient.
3. If there is a weight restriction on the number of  $-1$  and  $+1$ , accumulate all  $s_i$  securely, unmask the result and check whether the correct number of  $-1$  is hit. If not so, start over from Step 1.

Note that for NTRU, the initially sampled binary weight- $W$  polynomial is not rejected, but only the vector of signs. This adds  $\mathcal{B}(127, 254, 0.5)^{-1} \approx 20$  expected repetitions of the above procedure for NTRU-HPS2048{509, 677}, and for NTRU-HPS4096821  $\mathcal{B}(255, 510, 0.5)^{-1} \approx 28.3$ .

The numbers for sampling binary polynomials with correct weight are presented in Table 5, Table 6, and Table 7. It stands out that compared to the code-based schemes, a notably lower amount of randomness is required. This is due to the smaller polynomial degrees and the more favorable ratio between  $W$  and  $N$ . However, the very low numbers for NTRU are misleading, since they do not include the additional randomness required for sampling the correct sign weight.

#### 4.7 RepeatedAND

In [10], Drucker and Gueron propose ANDing random bit strings repeatedly with subsequent dedicated correction of the weight as a method for sampling fixed weight vectors. Starting with a zero bit string  $A$  of length  $N$ , they compute a random bit string  $\bar{A}$  of the same length by repeatedly ANDing random strings so that the expected weight of the string is halved with each AND, until the weight is below or equal to the target weight  $W$ . Then,  $A$  is set to  $A \vee \bar{A}$ , so that the new weight of  $A$  is less or equal the sum off the individual weights of  $A$  and  $\bar{A}$ . As long as the weight of  $A$  is not  $W$ , a new  $\bar{A}$  is computed with a target weight of the difference between the weight of  $A$  and  $W$  and ORed with  $A$  to increase its weight towards  $W$ .

Just like the simple rejection and our comparison sampling, this method is not secure for the decapsulation in HQC and BIKE.

At first sight, this method can be masked in a straight-forward manner, by checking the weight of secret intermediate vectors being the only non-trivial component. However, it makes heavy use of computing the (secret) weight of intermediate vectors, which is cheap in unmasked domain, but a big cost factor for masking. Experimentally, we found that for BIKE, HQC and McEliece, the average number of required weight checks significantly exceeds the average for our Comparison method presented in Sect. 4.6, with the smallest difference being McEliece-348864 (31.02 vs. 22.66), and the biggest difference being BIKE-L5 (60.38 vs. 30.30). In software, this masked weight check would predominantly determine the performance, rendering RepeatedAND obsolete for BIKE, HQC and McEliece. In hardware, however, the weight check could be performed in parallel with the ANDing. For NTRU, Streamlined NTRU Prime and NTRU LPRime the average number of comparisons are very similar, the RepeatedAND method, however, requires less randomness.

#### 4.8 Conversions Between Polynomial Representations

Some implementations of the cryptographic schemes use the index representation for fast multiplications that follow the sampling process, but one can also transform this representation to the coefficient representation in a constant-time and masked way. For each of the  $W$  non-zero indices one iterates over all coefficients  $N$  that are initialized with 0, and if the current indices are hit one replaces the 0 with a 1. We therefore need  $NW$  iterations with a `cmpeq` and a `cmov`. A conversion in the other direction from coefficient to index representation can be done similarly with comparable costs.

## 5 Masked Implementation

### 5.1 Software

We implemented all methods presented in Sect. 4 in software generalized for arbitrary masking order and thus secure against multi-trace power side-channel attacks. Except for the comparison method, our implementations are parametrized for  $N$  and  $W$ . We based our software implementations on masked gadgets presented in [7]. These gadgets can be proven to be secure under the assumptions provided by the  $d$ -probing model. Additionally, they fulfill certain composability notions ensuring that a design constructed by these gadgets is still secure in the  $d$ -probing model.

For Boolean masked software implementations, bitslicing is often a very efficient methodology to improve performance. All of our implementations are bitsliced as far as the algorithms allow, we also bitsliced all core operations presented in Sect. 4.1.

**Fisher-Yates.** The first component of the Fisher-Yates algorithm is to sample a random value with a varying range  $[0, N - i)$ . We implemented this in the additive masking domain with the biased multiplication method. To be compatible with unmasked implementations we take 32-bit Boolean masked randomness (for example from a masked Keccak) as input and transform it bitsliced to the 48-bit additive domain (modulo  $2^{48}$ ). We unbitslice the randomness and perform the multiplication with the public value  $N - i$  by a simple sharewise unmasked multiplication. The result is at most 48-bit wide, therefore the additive domain modulo  $2^{48}$  and not e.g.  $2^{64}$  which saves us some costly non-linear operations in the Boolean to arithmetic and arithmetic to Boolean conversions.

Taking the upper 16 bit from the results can be done in the Boolean domain, which we need anyway for the second component of the algorithm. But before transforming to the Boolean domain, we add  $i$  to the upper 16 bit, which is cheaper in the additive domain. The additive to Boolean transformation is again implemented bitsliced and we keep the data in the bitsliced domain for the comparisons and conditional moves of the second component. With  $W$  padded to the next multiple of 32, we can perform the inner loop with the comparison and condition move on 32 values at a time.

**Sorting.** To evaluate the sorting approach presented in Sect. 4.3 we implemented a masked bitonic sort in software. Bitonic sort for  $n$  elements performs  $n/2$  comparisons operating on all  $n$  elements in each iteration and each pair of elements that are compared has the same distance during one iteration. For the cases where the distance is greater than our register width of 32, we thus can directly compare and conditionally swap a group of 32 consecutive values with their respective pairs in the bitsliced domain where 32 values share a register for each bit.

Comparisons of elements with a distance of less than 32 are also possible in the bitsliced domain but require a transformation. When the distance is halved

from one iteration to the next as is the case for most iterations, we need to swap half of the bits of one group of 32 elements in one register with the respective other half of paired group. In the non-bit sliced domain, this would correspond to simple register swaps, in the bit sliced domain we need to swap bits by using rotations and Boolean operators. By implementing this transformation in the bit sliced domain, we are able to perform the entire sorting algorithm in the bit sliced domain and save transformations between the domains.

For distances below 32, our method works on 64 consecutive elements at a time, we therefore pad the polynomial width to the next multiple of 64 for a clear and efficient implementation. The additional coefficients appended by the padding get initialized with zero and not paired with random values, but with the highest value possible so that the nonzero lower coefficients will not be sorted to the additional indices and they can simply be cut off after sorting. Bitonic sort originally only works on power-of-two input sizes, but can be adapted to arbitrary sizes, as we did for our implementation.

**Rejection.** To be able to parallelize the comparison of  $r$  versus  $N$  we perform the outer loop on batches of 32 values. We then iterate over the batch, if the result of the comparison indicates that a value  $r$  is not less than  $N$  we directly continue with the next value. If not, we compare the value to the already sampled ones, again performing 32 bit sliced comparisons at a time. By performing 32 comparisons at a time we often perform comparisons with elements that are not yet set by the algorithm, but are initialized with a value e.g. zero. The result of these comparisons must not influence the rejection behavior, otherwise the initialization value can never be included in the output which would violate the uniform randomness requirement. We solve this by simply masking out the bits of these comparisons.

If no collision is found  $r$  is stored in the array, in contrast to the bounded rejection method, this condition is not a secret value, thus we do not need the masked `cmov` operation. But we implemented this move in the bit sliced domain, so that the array of indices can be kept in the bit sliced domain throughout the entire algorithm and only converted to the non-bit sliced domain at the end.

**Bounded Rejection.** Similar to the simple rejection method, the implementation of the bounded rejection method for the index representation has to deal with false collisions with the initialization values. Tracking which values are set and thus which comparisons are valid is cumbersome in this case because the amount of already correctly sampled values is secret. Instead, we implemented this by initializing the array with a value that is out of bound, e.g.  $N$ . This induces only a small overhead for the collision comparison, which now has to operate on  $\lceil \log_2(N) \rceil + 1$  bits instead of  $\lceil \log_2(N) \rceil$ .

Again we parallelized the inner loop with bit slicing to significantly improve the performance.

We determined the bound according to the formulas provided by Drucker et al. [15]. Drucker et al. suggest bounds for BIKE Level 1 ( $B = 327$ ) and Level 3

( $B = 488$ ) which give a probability to fail of less than  $2^{128}$ . If a sampling failure does not affect the security of the scheme, a lower bound can be chosen for better performance. We selected a bound of 704 for BIKE Level 5 and 364, 460 and 267 for the three relevant parameter sets of HQC to reach the same probability.

**Comparison.** Generally, this approach can be parallelized very efficiently as each coefficient is sampled individually. For software, this means that bitslicing is eligible, and for hardware, an individual trade-off between area and latency can be found.

In software, the weight check is the bottleneck of this method. Since it is hard to accumulate single masked bits in Boolean sharing on software platforms, we first deploy a bitsliced Boolean-to-additive masking conversion, which converts 32 masked bits to 32 arithmetically masked values modulo  $2^z$  for a sufficiently large chosen  $z$ . Then, we unbitslice these values and accumulate the additively masked values share-wise. Finally, when we iterated over the whole polynomial with this procedure, we can unmask the shared accumulation value to obtain the weight of the masked polynomial.

*Optimized Masked Weight Check.* To check whether the masked polynomial candidate has the correct weight or not, it is required to compute the weight of the polynomial. For this operation, the intermediate weight is a sensitive information as it could reveal the position of single coefficients. The masked weight computation itself is a secure accumulation of all masked coefficients to a value of size  $\lceil \log_2 N \rceil$  bits, e.g., by means of a secure  $\lceil \log_2 N \rceil$ -plus-one-bit adder. It is worth noting that for all three code-based applications, though,  $W$  is *much smaller* than  $N$ . It follows that most of the upper bits of such a secure adder are not required with overwhelming probability.

Since we know the expected weight of our polynomial candidate (under the assumption that no biased randomness is used as input), we can decrease the secure accumulator size and accept the possibility of an overflow happening. An overflow of the accumulator is not critical as long as it does not lead to a false-positive result, i.e., approving a polynomial that has not the correct weight.

Let  $z$  be the bit length of the secure accumulator output. Then, for a given  $(N, W, \ell, t)$  as explained in Sect. 4.6, we have probability  $p_{\text{fp}}$  of a false positive:

$$p_{\text{fp}} = \sum_{\substack{i = -\lfloor \frac{W}{2^z} \rfloor \\ i \neq 0}}^{\lfloor \frac{N-W}{2^z} \rfloor} \mathcal{B}\left(W + i \cdot 2^z, N, \frac{t}{2^\ell}\right) \quad (13)$$

Obviously, it is desirable to have a negligible  $p_{\text{fp}}$ , but also a low  $z$ , since this affects the efficiency of the weight check. We find that for all use cases and parameter sets, choosing  $z = 8$  (i.e., an 8-bit secure accumulator), yields  $p_{\text{fp}} < 2^{-200}$ .

**RepeatedAND.** Using the same weight check module as above, we also implemented the RepeatedAND method presented in [10]. This time, however, we cannot use the optimization shown above, since we do not check for equality, but rather whether a weight is bigger or smaller. Thus, we use 10-bit secure accumulation, which is enough for nearly all parameter sets of NTRU, Streamlined NTRU Prime, and NTRU LPRime. For Streamlined NTRU Prime- and NTRU LPRime-1277, the probability that an intermediate weight is greater or equal than  $2^{10}$  is negligible. This approach is not efficient for McEliece, BIKE and HQC, because compared to the Comparison approach, significantly more and bigger weight checks are required.

## 5.2 Hardware

As a case study, we implement the comparison sampling approach for BIKE in hardware. Additionally, we give some remarks on how hardware implementations of the other algorithms could be realized.

For hardware implementations, we generally have similar restrictions compared to embedded software platforms. Most importantly, only very limited memory is available, rendering sorting-based methods for high polynomial degrees infeasible. As an example, the smallest BIKE parameter set already would require  $32 \cdot 12323 \cdot 2 = 788672$  bit storage for first-order masking assuming that 31 bit randomness per coefficient would be sufficient. On the other hand, comparison-based sorting networks can be implemented very efficiently for smaller  $N$  as in NTRU and its variants and parallelized in a more fine-grained manner than for software platforms. This allows for precise trade-offs between latency and area demand.

To reduce the latency of comparisons, a parallel-prefix subtractor could be deployed by optimizing it to only obtain the uppermost carry-out bit. In return, this would require more secure non-linear gadgets compared to our comparison method presented in Sect. 4.1.

For Fisher-Yates, the boolean to arithmetic and vice versa transformations could be implemented with secure Boolean adders, which is possible efficiently and pipelined [17]. Then, the relatively big integer multiplications are a major cost factor in hardware, as they involve many bit operations.

For the RepeatedAND method, we certainly expect a higher control overhead due to the more complex algorithm compared to the comparison approach. Also, an intermediate masked vector must be stored in addition to the output vector, which results in a higher memory requirement. On the other hand, in contrast to software implementations, where the weight check is the bottleneck, we can execute the weight check in parallel to the secure AND operations. This could make this method efficient for the BIKE and HQC key generations and McEliece encapsulation.

**Comparison Method.** Since we aim for a masked implementation, we store each share of the target sampled polynomial in a separate memory (for BIKE

level 1, we instantiate one 18 KB memory for each share). Each of these memory modules can be accessed via a 32-bit interface. As explained in Sect. 4.6, the approach requires  $\ell$  bits of randomness to sample one bit. Due to the 32-bit interface of the memory modules, our hardware design samples 32bit in parallel which leads to  $\ell \cdot d \cdot 32$  bits of randomness required as input to the fixed input comparison. Since our target is to implement a side-channel resistant design, we replace all non-linear gates by secure gadgets (in our case study, we used Domain-Oriented Masking (DOM) gadgets [11]). As shown in Sect. 4.6, the comparison for BIKE level 1 consists of eight secure multiplication gadgets where each gadget requires  $\frac{d \cdot (d+1)}{2}$  bit of fresh randomness.

To track the Hamming weight of the sampled masked polynomial, we instantiate a masked Hamming weight computation unit. The design follows the implementation concept of the unmasked Hamming weight unit from [16]. However, we realize each adder stage by masked Ripple-Carry Adder (RCA) generated from HPC2 gadgets [8]. Eventually, we obtain a masked six-bit result for each 32-bit block which is fed into an accumulation stage. The accumulator is implemented by a fully pipelined masked 8-bit Sklansky adder as proposed in [2]. Since the adder consists of eight register stages, we obtain eight masked intermediate results that need to be accumulated to a final result. For this, we utilize the same adder and cleverly feed in the intermediate results from the adder to its input to add up all intermediate results. The final result is not secret and can be unmasked in order to compare it to the desired weight  $W$ . The procedure needs to be repeated in case the weight is not met.

## 6 Evaluation

### 6.1 Software

The target of our software implementations is the 32-bit Cortex-M4 microcontroller on the STM32F4 discovery board. To measure the cycle counts we set the frequency to 24MHz to make the cycle counts independent of the memory speed. We used the `arm-none-eabigcc-10.3.1` compiler with optimization-level 03 and report average cycle counts of 10 runs for algorithms without data-dependent branching and average counts of 1000 runs otherwise. For comparison sampling, we measure the non-branching execution of one iteration and report this value multiplied by the expected number of repetitions.

We excluded the generation of randomness required by calls to `rand` in our measurements so that only the performance of the fixed weight polynomial sampling algorithm is measured and not the performance of the PRNG. The generation of randomness required by masked operations is however included.

Tables 8 show our measurements in kilo cycle counts for first-order masking on the Cortex-M4.

From our measurements, we can first of all conclude that masked fixed-weight polynomial sampling is expensive in software.

For masked sampling of fixed-weight polynomials in the index representation, the simple rejection method is the fastest and can always be applied when there is



no need for seed security. If seed security is required, one could alternatively use the bounded method, which is always slower compared to the simple rejection. We thus only benchmarked the bounded rejection for the use cases in BIKE and HQC. Fisher-Yates also provides seed security, and outputs in the index representation and is faster than the bounded rejection for all parameter sets that we measured.

**Table 8.** Performance on the Cortex-M4 in kilo cycles for first order masking. Entries marked with – are irrelevant combinations that we did not implement/measure.

Scheme	$N$	$W$	Sort	Fisher-Y.	Reject	B. Reject	RepAND	Comp.	I2C Trans.
BIKE	24646	134	–	7128	–	34077	–	–	770708
BIKE	12323	71	–	2854	647	–	101629 <sup>a</sup>	45838	195945
BIKE	49318	199	–	13206	–	69140	–	–	2394245
BIKE	24659	103	–	4901	1255	–	156631 <sup>a</sup>	129050	592411
BIKE	81194	264	–	21680	–	131135	–	–	5497931
BIKE	40973	137	–	7514	2176	–	320522 <sup>a</sup>	234007	1372560
HQC	17669	75	–	3063	–	25803	–	–	309894
HQC	17669	66	–	2852	620	–	185348 <sup>a</sup>	63242	272707
HQC	35851	114	–	5377	–	41500	–	–	999526
HQC	35851	100	–	5034	1282	–	391503 <sup>a</sup>	183833	876778
HQC	57637	149	–	7808	–	28930	–	–	2094589
HQC	57637	131	–	7367	2132	–	837777 <sup>a</sup>	348099	1841552
McEliece	3488	64	108596	1847	462	–	19519 <sup>b</sup>	12948	32246
McEliece	4608	96	160777	3326	972	–	31778 <sup>b</sup>	20392	68236
McEliece	6688	128	240949	5044	1555	–	63766 <sup>b</sup>	31652	131539
McEliece	6960	119	249618	4848	1386	–	59875 <sup>b</sup>	34571	127568
McEliece	8192	128	300713	4591	1527	–	62867 <sup>b</sup>	34609	161312
NTRU	509	254	9699	11532	4709	–	2141	1666	15342
NTRU	677	254	14958	12445	4833	–	3559	2935	22674
NTRU	821	510	18338	17737	7022	–	4140	3921	32655
sNTRU Prime	653	288	14958	15086	6345	–	3023	3033	24650
NTRU LPRime	653	252	14958	12390	4806	–	3177	3299	21515
sNTRU Prime	761	286	16464	15063	6005	–	3699	3336	27828
NTRU LPRime	761	250	16464	12350	4570	–	3457	3773	24264
sNTRU Prime	857	322	19848	20249	7461	–	4125	4012	34948
NTRU LPRime	857	329	19848	20569	7805	–	4482	4650	35825
sNTRU Prime	953	396	21564	27494	11253	–	4403	6266	47664
NTRU LPRime	953	345	21564	20867	8404	–	4496	6617	41385
sNTRU Prime	1013	448	23405	31680	14421	–	4836	5763	57395
NTRU LPRime	1013	392	23405	27380	10843	–	5428	7015	50228
sNTRU Prime	1277	492	32361	42388	18445	–	6861	9612	85013
NTRU LPRime	1277	429	32361	33673	13822	–	7285	9245	74318

<sup>a</sup> average over 3 executions <sup>b</sup> average over 10 executions

The sorting method required too much stack to fit into the 192-KB SRAM of our board for the large  $N$  of BIKE and HQC, but the results of McEliece with medium sized  $N$  already indicate high costs for large  $N$ . As the runtime for sorting grows sub-quadratic in  $N$ , but Fisher-Yates performance is mainly determined by its  $\mathcal{O}(W^2)$  loop iterations, the sorting method is faster for the higher parameter sets of Streamlined NTRU Prime and NTRU LPRime which have medium sized  $N$  and relatively high  $W$ .

However, sorting is always outperformed by the two other coefficient representation methods, the RepeatedAND and our comparison method. For BIKE, HQC and McEliece the comparison method is superior to the RepeatedAND in runtime costs. For NTRU the performance of both methods is very similar, for Streamlined NTRU Prime and NTRU LPRime, RepeatedAND is mostly faster.

In the last column of Table 8 we present the cycle counts for a masked transformation from index to coefficient representation. In general, it depends on the implementation of the scheme which representation is required for further operations, the index representation of sparse polynomials can for example be used for efficient multiplications. The high costs for a masked transformation indicate, that if only a single representation is required, a method that directly outputs the correct representation is usually preferable.

To summarize the recommendations for masked software implementations derived from our performance measurements: if no seed security is required, the simple rejection method is the fastest index sampling method and either RepeatedAND or our comparison method is preferable for an output in coefficient representation. For the scenarios in BIKE and HQC, where the seed must be kept secret, Fisher-Yates yields the best performance with an output in index representation, and one could use a masked transformation to get the coefficient representation, or implement a less memory-consuming variant of the sorting method, by utilizing radixsort, as explained in Sect. 4.3.

Two schemes have parameter sets that lead to a special case for some algorithms. When  $N$  is a power of two which is the case for one parameter set of McEliece, then Fisher-Yates and rejection sampling become easier because checking if  $r < N$  is not necessary. We adopted our code accordingly when benchmarking this parameter set and the effect shows clearly in the cycle numbers of Fisher-Yates that are lower compared to the next smaller parameter set of McEliece. The second special case is the highest parameter set for NTRU where  $W > N/2$ . In this case, the symmetry of fixed weight sampling allows to sample with  $W' = N - W$  instead.

## 6.2 Hardware

Table 9 shows the hardware implementation results for the comparison approach presented in Sect. 4.6 for BIKE level 1. Therefore, we implement our design for a Xilinx Artix-7 xc7a200 Field-Programmable Gate Array (FPGA) and report the required resources and performance numbers. As a baseline, we first implement an unprotected design that consumes just 100 slices and finishes on average one sampling process in 33.4  $\mu$ s. Note, the number of required Block-RAMs (BRAMs) is reported in 36 KB memory modules. Therefore, the unprotected design requires only one 18 KB memory to store the final polynomial.

**Table 9.** Implementation results for the comparison sampling approach for BIKE level 1 on an Artix-7 FPGA.

$d$	Resources				Performance		
	Logic		Memory	Area	Cycles	Frequency	Latency
	LUT	FF	BRAM	Slices	Cycles	MHz	$\mu s$
0	194	115	0.5	100	8 350	250	33.400
1	1 957	2 721	1	627	9 756	250	39.024
2	5 075	5 815	1.5	1 548	9 756	250	39.024
3	9 038	10 085	2	2 584	9 756	250	39.024

The next three lines in Table 9 show the implementation results for a first, second, and third-order protected design. The first-order protected implementation consumes 627 slices compared to 100 slices of the unprotected design. However, all protected implementations of the sampler can be executed with the same frequency, but have a slightly higher latency due to additional register stages introduced by the masking approach.

## 7 Conclusion

In this work, we demonstrated how all fixed-weight polynomial sampling methods in the literature can be masked at arbitrary order. Our implementations indicate that despite bitslicing and optimized subcomponents, the existing algorithms are costly for masked software. Drucker and Gueron [10] benchmarked a subset of our algorithms and schemes without power side-channel countermeasures, their numbers indicate that the relative performance of the sampling algorithms for a given scheme is equal for masked and nonmasked software implementations.

The flexibility of hardware implementations allows faster solutions, further implementations would be an interesting target for future work. Additionally, we identified that shuffling should be investigated for the sampling algorithms as an efficient countermeasure against single-trace attacks.

**Acknowledgments.** The work described in this paper has been supported by the German Federal Ministry of Education and Research BMBF through the project QuantumRISC (16KIS1038) and PQC4Med (16KIS1044), the German Research Foundation DFG under Germany’s Excellence Strategy - EXC 2092 CASA - 390781972 and the European Commission under the grant agreement number 101070374. We thank Eike Kiltz and Gregor Leander for their valuable comments.

## References

1. Azouaoui, M., et al.: Leveling Dilithium against leakage: revisited sensitivity analysis and improved implementations. Cryptology ePrint Archive, Paper 2022/1406 (2022). <https://eprint.iacr.org/2022/1406>

2. Bache, F., Güneysu, T.: Boolean masking for arithmetic additions at arbitrary order in hardware. *Appl. Sci.* **12**(5), 2274 (2022)
3. Barthe, G., et al.: Masking the GLP lattice-based signature scheme at any order. In: Nielsen, J.B., Rijmen, V. (eds.) *EUROCRYPT 2018*. LNCS, vol. 10821, pp. 354–384. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78375-8\\_12](https://doi.org/10.1007/978-3-319-78375-8_12)
4. Batcher, K.E.: Sorting networks and their applications. In: *AFIPS Conference*, vol. 32, pp. 307–314. Thomson Book Company, Washington D.C. (1968)
5. Bernstein, D.J.: Divergence bounds for random fixed-weight vectors obtained by sorting (2020)
6. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU prime: reducing attack surface at low cost. In: Adams, C., Camenisch, J. (eds.) *SAC 2017*. LNCS, vol. 10719, pp. 235–260. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-72565-9\\_12](https://doi.org/10.1007/978-3-319-72565-9_12)
7. Bronchain, O., Cassiers, G.: Bitslicing Arithmetic/Boolean masking conversions for fun and profit with application to lattice-based KEMs. *IACR Trans. Crypt. Hardware Embed. Syst.* **2022**(4), 553–588 (2022)
8. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.-X.: Hardware private circuits: from trivial composition to full verification. *IEEE Trans. Comput.* **70**(10), 1677–1690 (2021)
9. Coron, J.-S., Gérard, F., Trannoy, M., Zeitoun, R.: High-order masking of NTRU. *Cryptology ePrint Archive*, Report 2022/1188 (2022). <https://eprint.iacr.org/2022/1188>
10. Drucker, N., Gueron, S.: Generating a random string with a fixed weight. In: Dolev, S., Hendl, D., Lodha, S., Yung, M. (eds.) *CSCML 2019*. LNCS, vol. 11527, pp. 141–155. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-20951-3\\_13](https://doi.org/10.1007/978-3-030-20951-3_13)
11. Groß, H., Mangard, S., Korak, T.: Domain-oriented masking: compact masked hardware implementations with arbitrary protection order. In: *TIS@CCS*, p. 3. ACM (2016)
12. Guo, Q., Hlauschek, C., Johansson, T., Lahr, N., Nilsson, A., Schröder, R.L.: Don't reject this: key-recovery timing attacks due to rejection-sampling in HQC and BIKE. *IACR Trans. Crypt. Hardware Embed. Syst.* **2022**(3), 223–263 (2022)
13. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) *CRYPTO 2003*. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45146-4\\_27](https://doi.org/10.1007/978-3-540-45146-4_27)
14. Karabulut, E., Alkim, E., Aysu, A.: Single-trace side-channel attacks on  $\omega$ -small polynomial sampling: with applications to NTRU, NTRU prime, and CRYSTALS-DILITHIUM. In: *IEEE HOST*, pp. 35–45. IEEE (2021)
15. Kostic, D., Drucker, N., Gueron, S.: Isochronous implementation of the errors-vector generation of BIKE (2022). <https://github.com/awslabs/bike-kem>. Accessed 25 Oct 2022
16. Richter-Brockmann, J., Mono, J., Güneysu, T.: Folding BIKE: scalable hardware implementation for reconfigurable devices. *IEEE Trans. Comput.* **71**(5), 1204–1215 (2022)
17. Schneider, T., Moradi, A., Güneysu, T.: Arithmetic addition over boolean masking. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) *ACNS 2015*. LNCS, vol. 9092, pp. 559–578. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-28166-7\\_27](https://doi.org/10.1007/978-3-319-28166-7_27)
18. Sendrier, N.: Secure sampling of constant-weight words - application to BIKE. *Cryptology ePrint Archive*, Report 2021/1631 (2021). <https://eprint.iacr.org/2021/1631>

**MPC**



# Private Polynomial Commitments and Applications to MPC

Rishabh Bhadauria<sup>1</sup>(✉), Carmit Hazay<sup>1,3</sup>,  
Muthuramakrishnan Venkatasubramanian<sup>2,3</sup>, Wenxuan Wu<sup>4</sup>,  
and Yupeng Zhang<sup>4</sup>

<sup>1</sup> Bar Ilan University, Ramat Gan, Israel  
rishabh.bhadauria@biu.ac.il

<sup>2</sup> Georgetown University, Washington, DC, USA

<sup>3</sup> Ligerio Inc., Rochester, USA

<sup>4</sup> Texas A&M University, College Station, USA

**Abstract.** Polynomial commitment schemes allow a prover to commit to a polynomial and later reveal the evaluation of the polynomial on an arbitrary point along with proof of validity. This object is central in the design of many cryptographic schemes such as zero-knowledge proofs and verifiable secret sharing. In the standard definition, the polynomial is known to the prover whereas the evaluation points are not private. In this paper, we put forward the notion of *private polynomial commitments* that capture additional privacy guarantees, where the evaluation points are hidden from the verifier while the polynomial is hidden from both.

We provide concretely efficient constructions that allow simultaneously batch the verification of many evaluations with a small additive overhead. As an application, we design a new concretely efficient multi-party private set-intersection with malicious security and improved asymptotic communication and space complexities.

We demonstrate the concrete efficiency of our construction via an implementation. Our scheme can prove  $2^{10}$  evaluations of a private polynomial of degree  $2^{10}$  in 157 s. The proof size is only 169 KB and the verification time is 11.8 s. Moreover, we also implemented the multi-party private set intersection protocol and scale it to 1000 parties (which has not been shown before). The total running time for  $2^{14}$  elements per party is 2,410 s. While existing protocols offer better computational complexity, our scheme offers significantly smaller communication and better scalability (in the number of parties) owing to better memory usage.

## 1 Introduction

A polynomial commitment is a cryptographic building block that allows a prover to commit to a polynomial, which can later be opened at any evaluation point with proof that the evaluation is correctly computed. Polynomial commitments, which serve as an important building block in constructing cryptographic protocols, were introduced by Kate et al. [46] for the construction of verifiable secret sharing in the synchronous and asynchronous setting [6].

The scheme was generalized to multivariate polynomials by Papamanthou et al. [51], and to zero-knowledge proofs of knowledge by Zhang et al. [69]. In recent years, they are extensively used to build efficient zero-knowledge proof systems [23, 32, 59, 62, 65, 68], where recent new schemes without a trusted setup were proposed in [15, 47, 61, 62, 67]. Subsequent works considered batched openings for multiple evaluations [60] and multiple polynomials [37]. Another application where polynomial commitments are utilized is “Proof of retrievability” [45, 66]. In this problem, the server wishes to prove to a verifier that all of the client’s data is stored correctly. The polynomial commitment allow the prover to prove the integrity of the data storage. Logarithmic and constant size polynomial commitments are also used in constructing vector commitments [17, 18, 22].

To date, all concretely efficient polynomial commitments require the verifier to know the evaluation point and the prover to know the polynomial. While such a notion is sufficient to design succinct zero-knowledge arguments, secure multi-party computation (MPC) requires additional privacy guarantees. In this paper, we consider a different setting where the polynomial is unknown to the prover and is encrypted. Moreover, the evaluation points are committed by the prover and may not be publicly known to the verifier. This setting is very common in MPC where both the polynomial and the evaluation points must remain private as they are defined based on the parties’ inputs. We denote this primitive by *private polynomial commitment* and show that it can be used as a building block in many applications that arise in the secure multi-party setting; see Sects. 1.1 and 4. Our scheme is particularly useful in batch scenarios when there are multiple evaluation points. In this case, the proof size and verifier’s complexity grow additively with the number of points.

## 1.1 Our Contributions

Our contribution is threefold. (1) abstracting the new notion of private polynomial commitments and providing two constructions. (2) demonstrating its applicability for MPC and (3) implementing our commitment schemes and presenting a new multi-party private set-intersection (MPSI) protocol.

**Private Polynomial Commitments.** Our contribution includes two flavors of private polynomial commitments with a hidden (encrypted) polynomial; one where the evaluation points are public and the other where they are private. Our schemes are built on the recent scheme of an inner product argument [16], which generalizes the inner product argument from [14] to bilinear groups. Specifically, we embed the ciphertexts encrypting the coefficients in the base group using an Additively Homomorphic Encryption (AHE) scheme introduced in [13]. Working with bilinear maps allow to publicly verify a single multiplication in the exponent which allows any party to verify the proof. More specifically, for a polynomial of degree  $d$ , the overhead is dominated by  $O(d)$  bilinear pairings whereas the proof size is  $O(\log d)$  and the verifier time is  $O(d)$  exponentiations. Our construction supports batched evaluations efficiently. To open at  $m$  evaluation points, the proof size is  $O(m + \log d)$  and the verifier time is only  $O(m + d)$ . The polynomial is hidden from all parties and only an encrypted form is available to the prover.

Our constructions rely on two different commitment schemes for committing to the encrypted polynomial (using the pairing-based scheme from [4]) and the evaluation points (using Pedersen commitment [52]). We further rely on the Boneh et al. pairing-based encryption scheme [13] to be compatible with our pairing-based commitment scheme, both of which rely on the Decisional Linear Assumption (DLIN) and Double Pairing Problem (DPP).

Our commitment scheme uses an inner product argument [14] as a building block (denoted by BBB-IPA) and is the first polynomial commitment scheme where the prover does not know the actual polynomial and only has access to its encryption. The main challenge in constructing this commitment scheme was the integration of encrypted polynomials into the polynomial commitment scheme. Secondly, directly constructing a scheme would not provide batching. To ensure batching and overall small proof size, we reduce the proving of the polynomial evaluation to multiple inner products. First, we provide a new inner product argument that allows the prover to verify inner products on encrypted ciphertext with the evaluation vector. Second, we prove the correct structure of multiple evaluation vectors by verifying the linear and quadratic constraints. Both our linear and quadratic tests reduce the multiple constraints on all different evaluation vectors to verify a single inner product argument, thereby ensuring the batching feature is effective. An additional feature is that the proof can be made non-interactive using Fiat-Shamir.

**Applications.** Private polynomial commitment schemes are useful for private computations based on polynomials. We list four such applications that can benefit from the scalability and batching of the evaluations as inherent in our commitment scheme. Firstly, we use our new private polynomial commitment as a building block to present a new scalable multi-party PSI protocol that is secure against malicious adversaries. We also discuss three other applications - Oblivious Polynomial Evaluation, Verifiable Polynomial Evaluation, and Non-Interactive two-party PSI; for more details see Sect. 4.

**Scalable Multi-party Private Set-Intersection (MPSI).** PSI is a fundamental problem in secure computation that has been widely studied in the past decade. In this problem a set of parties  $P_1, \dots, P_n$ , holding input sets  $X_1, \dots, X_n$  of sizes  $m_1, \dots, m_n$ , respectively, wish to compute  $X_1 \cap X_2 \cap \dots \cap X_n$ . The two-party setting has been studied extensively and continues to be a hot topic of research owing to numerous applications such as contact discovery, dating services, data mining, recommendation systems, and law enforcement. In a long line of works, highly efficient two-party protocols have been designed with almost linear overhead in the set sizes (see some recent works at [21, 53–55] and references therein). Furthermore, Google has recently leveraged this technology to match login credentials against an encrypted database.

While considerable progress has been made in the two-party setting, very few works have explored the concrete efficiency of PSI in the multi-party setting and



the existing works have mostly considered only the semi-honest setting. Furthermore, current approaches fail to achieve overheads as in the two-party setting and do not scale well due to communication and space bottlenecks. Multiparty PSI is a fundamental cryptographic primitive with a richer set of applications beyond the two-party ones such as distributed intrusion detection, identifying the most visited sites or watched movies, contact tracing and more.

Our starting point is the work of Freedman et al. [31] who designed a simple two-party PSI protocol based on polynomials. Roughly speaking,  $P_1$  creates a polynomial  $Q(\cdot)$  whose roots correspond to its input data set and sends this polynomial to  $P_2$ , encrypted under an additively homomorphic encryption scheme.  $P_2$  homomorphically evaluates a “masked” variant of the encrypted polynomial on its data set. In more detail, for each element  $x$  in  $P_2$ ’s input set,  $P_2$  generates fresh randomness  $r$  and sends an encryption of  $r \cdot Q(x) + x$  to  $P_1$ .  $P_1$  decrypts and identifies the elements in the set intersection. Namely, if the decrypted value  $x$  is in  $P_1$ ’s set, then  $x$  is extracted from the decryption of the ciphertext. Whereas if the item  $x$  is not in the intersection, with very low probability, there exists an element  $z$  for which  $r \cdot Q(z) + z$  is a false positive.

More recently, Hazay and Venkatasubramanian [43] extended [31] to the multi-party setting by reducing the multi-party PSI (MPSI) task among  $n$  parties to  $n$  instances of two-party PSI. In this work we explore the practicality of [43] in the malicious setting where up to  $n - 1$  parties can be corrupted. On a high level, in [43], parties  $P_2, \dots, P_n$  create a polynomial whose roots correspond to their respective inputs and send their encrypted coefficients to  $P_1$ .  $P_1$  then aggregates the polynomials and homomorphically evaluates the resulting encrypted polynomial on its input set. To make the protocol secure against malicious adversaries, [43] introduced a simple mechanism for  $P_1$  to prove and the parties to verify that  $P_1$  aggregated the polynomials correctly, and relied on zero-knowledge proofs for the remaining steps.

The protocol presented in [43] implies an overall communication complexity of  $O(n^2 + n \cdot m_{\max} + n \cdot m_{\min} \cdot \log m_{\max})$  where  $m_{\max}$  (resp.  $m_{\min}$ ) is the size of the largest (resp. smallest) input set. The threshold key generation incurs a communication cost of  $O(n^2)$ . The central party aggregates the input polynomials of all the parties and returns the encrypted coefficients of the aggregated polynomial. This yields a communication overhead of  $O(n \cdot m_{\max})$ . The main source of overhead is due to the zero-knowledge proof applied by the central party for proving correct evaluation, which implies an overhead of  $O(n \cdot m_{\min} \cdot \log m_{\max})$ . This phase is captured in our protocol by private polynomial commitments.

More precisely, in this work, we introduce a variant of [43] where we rely on a new abstraction that is based on private polynomial commitments. By leveraging the efficiency and batching features of our commitment schemes, we manage to improve the communication and computation complexities of [43]. We further provide an implementation of our PSI protocol and explore its concrete efficiency. This is in contrast to [43] that had the potential of being concretely efficient but did not provide an implementation.

THE COMPLEXITY OF OUR PROTOCOL. In addition to our new abstraction, we further improve the asymptotic complexity of [43] to  $O(n^2 + \sum_{i=1}^n m_i + n \cdot (m_{\min} +$

$\log m_{\max}$ ). Introducing private polynomial commitments (PPC) as a building block, the central party in our protocol does not send the encrypted aggregated polynomial. Instead, a commitment of encrypted aggregated polynomials is sent to the parties. This allows us to remove the  $O(n \cdot m_{\max})$  factor. To further reduce the communication complexity, we leverage the batching feature of PPC which allows the central party to prove the correctness of multiple evaluations on the aggregated polynomial. The proof size, in this case, is  $O(m_{\min} + \log m_{\max})$  which contributes an additive factor of  $O(n \cdot (m_{\min} + \log m_{\max}))$  to the communication complexity of our MPSI protocol. A detailed analysis is provided in Table 3 where the communication complexity is broken according to the central party overhead and the other parties and is presented for each phase separately.

COMPARISON WITH RECENT WORK. Three recent works that design PSI protocols with malicious security are [9, 33, 38]. Similarly to our work, these works also achieve linear communication complexity in the number of parties by relying on a star topology. The main advantage of these protocols is that they rely on oblivious transfer (OT), oblivious linear evaluation (OLE) (used in [38]) and symmetric-key primitives for which we have very efficient instantiations. In comparison to previous work [9, 33, 38], our protocol achieves the best communication and space complexities. Specifically, our communication complexity is dominated by the term  $O(n^2\kappa + nm\kappa)$  where the gain compared to previous work is due to an aggregation of the encrypted input polynomials and the small batched proof size. We compare the communication complexity in Table 1. In the typical parameter regime, the computational security parameter  $\kappa$  is greater than the statistical parameter  $\lambda$  satisfying the inequality  $\lambda + \log m < \kappa$  where  $m$  is the input set size. Applying this inequality to the asymptotic communication complexity of [33] yields communication complexity that matches ours.

Most MPSI protocols (including ours) are designed for a star topology, where a central party aggregates the other parties' messages and therefore requires larger space. In prior works, the space complexity of the central party is inflated with a factor that depends both on the input and the number of parties, whereas our space complexity only grows with  $O(m\kappa)$ . The space complexity of the other, "non-central" parties, is independent of the number of parties. We compare the space complexity in Table 2.

Our paper realizes a standard MPSI functionality where a single party (typically the central party) receives the output, but can be extended to guarantee security even when all parties receive the output. Both [38] and our protocol achieve this standard security whereas the works of [9, 33] provide a weaker security guarantee that allows the party that first receives the output (if controlled by the adversary) to unnoticeably remove certain elements from the output when broadcasting it to all parties. Note that these protocols can achieve full security, but this will require applying general-purpose zero-knowledge proofs.

On the other hand, the computational cost of [9, 33, 38] grows with  $\Omega(mn\kappa)$  field multiplications, while the dominating cost of our computation is  $O(m^2)$  exponentiations. This can be further reduced into  $O(m \frac{\log m}{\log \log m})$  using hashing. While for a small number of parties, our protocol is slower, the total running

**Table 1.** The communication complexity analysis of MPSI *in bits* where  $\kappa$  is the computational security parameter,  $\lambda$  is the statistical security parameter,  $n$  is the number of parties,  $m$  is an upper bound on the inputs set sizes and  $P_1$  is the central party.

	$P_1$	$P_i$	Total
[9]	$O(nm\kappa^2 + nm\kappa \log m\kappa)$	$O(m\kappa^2 + m\kappa \log m\kappa)$	$O(nm\kappa^2 + nm\kappa \log m\kappa)$
[33]	$O(n\kappa + nm(\kappa + \lambda + \log m))$	$O(n\kappa + m(\kappa + \lambda + \log m))$	$O(n^2\kappa + nm(\kappa + \lambda + \log m))$
[38]	$O(nm\kappa + n\lambda\kappa \log m)$	$O((n + m)\kappa + \lambda\kappa \log m)$	$O(n^2\kappa + nm\kappa + n\lambda\kappa \log m)$
Theorem 2	$O(nm\kappa)$	$O((n + m)\kappa)$	$O(n^2\kappa + nm\kappa)$

time essentially remains the same when the number of parties increases. For instance, our experiments show that our scheme takes 9,141s for 1000 parties and  $2^{16}$  elements per party. Prior works cannot run at this scale.

We highlight some applications which require PSI for a large number of parties and large input sizes: (1) Cache-sharing [50] involves multiple network providers who wish to cache common elements with high access frequency in a shared cache and require privacy of their local cache. (2) Another application is to generate statistics over the Tor network. Prior literature e.g., [26, 63] has relied on MPC, secure aggregation and differential privacy to generate statistics on Tor servers in a privacy-preserving manner. Large-scale MPSI can be useful here where common features need to be extracted among the relay servers without compromising the users’ privacy. (3) Hospitals and healthcare providers can collaborate to analyze common features between databases which include a large number of medical records. (4) Finally, MPSI can be applied for contact tracing. A large group of patients can execute an MPSI protocol to find common locations they have been to without leaking each individual’s travel history. The result can help the actions of testing or quarantine in these areas.

**Table 2.** The space complexity analysis of MPSI *in bits* where  $\kappa$  is the computational security parameter,  $n$  is the number of parties,  $m$  is an upper bound on the inputs set sizes and  $P_1$  is the central party.

	$P_1$	$P_i$
[9]	$O(nm\kappa^2 + m\kappa \log m\kappa)$	$O(m\kappa^2)$
[33]	$O(nm\kappa + m(\kappa + \lambda + \log m))$	$O(m(\kappa + \lambda + \log m))$
[38]	$O(nm\kappa)$	$O(m\kappa)$
Theorem 2	$O(m\kappa)$	$O(m\kappa)$

Private polynomial commitments are also useful for reusable non-interactive two-party PSI. Non-interactive secure computation introduced in [44], considers a “receiver” that publicly broadcasts a single message and any “sender” can interact in a two-party secure computation protocol with the receiver by sending

a single message to the receiver. The receiver only needs to broadcast once and any number of interactions with the receiver can be performed. Specializing the setting to PSI, our protocol enables non-interactive PSI which can be applied to dating services, ride-share matching, and contact tracing. While such a protocol may introduce high computational cost compared to existing works e.g., [57], its communication cost is competitive as it benefits from our batching feature, which is extremely useful in a client-server setting; see more details in Sect. 4.3.

**Oblivious Polynomial Evaluation.** The oblivious polynomial evaluation (OPE) functionality is an important functionality in the field of secure two-party computation. It considers a setting where party  $P_2$  holds a  $d$ -degree polynomial  $Q(\cdot)$  and party  $P_1$  holds an element  $t$ , and the goal is that  $P_1$  obtains  $Q(t)$  and nothing else while  $P_2$  learns nothing. OPE has proven to be a useful building block and can be used to solve numerous cryptographic problems; e.g., secure equality of strings, set-intersection, approximation of a Taylor series, RSA key generation, oblivious keyword search, set membership, blacklisting anonymous users, data entanglement and more [8, 30, 31, 35, 40, 49].

In this work, we consider a distributed variant of OPE, where the input polynomial is additively secret shared amongst the parties, and the goal of the parties is to evaluate (in the exponent) the aggregated polynomial privately and correctly. The scenario where the polynomial is distributed naturally arises in settings where the data cannot be stored on a single memory device due to privacy considerations. Secret-sharing sensitive data protects it against leakage attacks and eliminates the risk of breaching the stored memory. In some cases, the data is distributed in order to avoid a single point of failure and to ensure continuous access to the data.

Private polynomial commitments are useful in this context and enable secure evaluation of the combined polynomial in the presence of  $n - 1$  malicious corruptions, similar to our PSI protocol. The ingoing communication complexity of  $P_1$  is linear in the size of shares, whereas the outgoing communication only grows logarithmically in the polynomial degree plus  $P_1$ 's input size (and hence sublinear in  $d$ ). The bulk of the computational overhead is attributed to  $P_1$ , which evaluates the aggregated polynomial on its input. An interesting feature of our protocol is its usage for multi-point evaluations. Here  $P_1$  evaluates  $Q(\cdot)$  on multiple points  $t_1, t_2, \dots$  where the accumulated overhead per evaluation point for ensuring malicious security vanishes away due to our batching property.

**Verifiable Polynomial Evaluations.** In this setting, computationally weak devices (or clients) wish to outsource their computation and data to an *untrusted* server in the cloud. The ultimate goal in this setting is to design efficient protocols that minimize the computational overhead of the clients and instead rely on the extended resources of the server. Of course, the amount of work invested by the client for verifying the correctness of the computation is *substantially* smaller than running the computation by itself. Another ambitious challenge of verifiable computation is to minimize the *communication* from the cloud.

The problem of delegating a single polynomial was studied by Benabbas et al. [10], who introduced a new cryptographic primitive of algebraic PRFs, which enables the generation of short authentication message to verify the server’s reply. Followup works [7, 19, 20, 27] improved different aspects of [10]. Nevertheless, all prior constructions considered a setting where a single client communicates with the server. Extending these solutions to the multi-client setting is not immediate (even in the non-private setting) since the server needs to aggregate the shares of the polynomials and provide proof for validating the aggregation, which is highly non-trivial. We observe that polynomial commitment schemes directly imply a verifiable evaluation of distributed polynomials where correctness is established via the proof provided by the server.

When considering verifiable computation, one can consider a setting where the function is either public or private. Verifiable computation with function privacy is often harder to achieve. We note that our construction follows even if the polynomials are encrypted while the evaluation points are given in the clear. This can capture scenarios where the polynomial represents a database with secret payloads yet the queries are not private.

**Implementation Details.** To validate the concrete efficiency of our construction, we implemented our private polynomial commitment scheme and multi-party PSI protocol. Our implementation of the private polynomial commitment scheme demonstrates the advantage of the batch opening. For a polynomial of degree  $2^{16}$ , the proof size is 18.6 KB and the verifier time is 53.7 s to open one evaluation, while they are only 6.1 MB and 757 s for  $2^{16}$  evaluations respectively, which are significantly better than repeating the single opening  $2^{16}$  times. Our multi-party PSI protocol with malicious security can scale to 1000 parties with  $2^{16}$  elements per party. The majority of the time is spent on the computation of the proofs of our private polynomial commitment, which can be further accelerated through multi-threading and hashing. The communication and the memory usage of our protocol is an order of magnitude better than existing schemes, and thus our protocol performs better for a large number of parties and networks with limited bandwidth; see Sect. 5 for further details. We plan to open-source our implementation and the source code is available at <https://anonymous.4open.science/r/PCOM-CCF4>.

## 2 Private Polynomial Commitment Schemes

In this section, we introduce a new polynomial commitment scheme with privacy features. Loosely speaking, such a protocol is carried out between a committer  $C$  and a receiver  $R$  where  $C$  commits to an encrypted polynomial  $\mathbf{C}$ , denoted by a sequence of ciphertexts  $\mathbf{C} = (c_0, c_1, \dots, c_d)$  where  $c_i$  is a ciphertext that encrypts the  $i^{\text{th}}$  coefficient of the underlying plaintext polynomial. In these schemes, upon committing to the encrypted polynomial,  $C$  sends  $\mathbf{C}$  to  $R$  and later evaluates it at an evaluation point  $t$ . Following that,  $C$  proves that a ciphertext  $c_y$  is a correct evaluation of the encrypted polynomial at some private evaluation point  $t$ .

## 2.1 Security Definitions

We continue with the security definition of our new polynomial commitments.

**Definition 1.** (*Private Polynomial Commitments with Hidden Evaluation Points*) Let  $E = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval}, \text{Rerand})$  be an AHE scheme with groups  $\mathcal{M}$  and  $\mathcal{C}$ . Let  $PK$  be the public key of the underlying AHE scheme and generated by  $E.\text{KeyGen}$ . A private commitment scheme PCOM w.r.t  $E$  is a tuple of algorithms  $(\text{Setup}, \text{Commit}, \text{CommitPt})$  and a protocol  $(C, R)$  defined as follows:

- $\text{pp} \leftarrow \text{Setup}(1^\kappa, d)$ : takes an input  $\kappa, d$  where  $\kappa$  is the security parameter and  $d$  is the degree of the polynomial, and outputs public parameters  $\text{pp}$ .
- $\text{com}_{\mathbf{C}} \leftarrow \text{Commit}(\text{pp}, \mathbf{C}; r_{\mathbf{C}})$ : takes as input a public parameters  $\text{pp}$ , a vector of ciphertexts (representing an encrypted polynomial)  $\mathbf{C} = (c_0, c_1, \dots, c_d)$  where  $c_i \in \mathcal{C}$  for all  $i$  and randomness  $r_{\mathbf{C}}$ , and outputs a commitment  $\text{com}_{\mathbf{C}}$ .
- $\text{com}_T \leftarrow \text{CommitPt}(\text{pp}, t, d; r_T)$ : takes as input public parameters  $\text{pp}$ , an evaluation point  $t$ , a randomness  $r_T$  and  $d$  is the degree of the polynomial and outputs a commitment  $\text{com}_T$ .
- $(C, R)$  is a public-coin interactive protocol between  $C$  and  $R$ . Both  $C$  and  $R$  have common inputs, public parameters  $\text{pp}$ , a public-key  $PK$  for the underlying AHE scheme, a commitment  $\text{com}_{\mathbf{C}}$ , another commitment  $\text{com}_T$  and an evaluation ciphertext  $c_y \in \mathcal{C}$ .  $C$  additionally receives as input an encrypted polynomial  $\mathbf{C}$ , an evaluation point  $t$  and randomness  $r_{\mathbf{C}}, r_{c_y}, r_t$ . At the end of the protocol execution,  $R$  either outputs accept or reject. We denote by  $(C(\mathbf{C}, t, r_{\mathbf{C}}, r_{c_y}, r_T), R)$   $(\text{pp}, PK, \text{ck}, \text{com}_{\mathbf{C}}, \text{com}_T, c_y)$  the random variable representing an execution and given an instance of the execution  $e$ , we denote by  $\text{view}_1(e)$  (resp.  $\text{view}_2(e)$ ) the view of the  $C$  (resp.,  $R$ ) and  $\text{out}_1(e)$  (resp.,  $\text{out}_2(e)$ ) the output of  $C$  (resp.,  $R$ ).

We require the following security properties to be satisfied:

**Completeness:** For any vector of ciphertexts  $\mathbf{C} = (c_0, c_1, \dots, c_d)$  generated using  $PK \leftarrow E.\text{KeyGen}(1^\kappa)$  and an evaluation point  $t$ , we have that:

$$\Pr \left[ \begin{array}{l} \text{pp} \leftarrow \text{PCOM.Setup}(1^\kappa, d); \\ \text{com}_{\mathbf{C}} \leftarrow \text{PCOM.Commit}(\text{pp}, \mathbf{C}; r_{\mathbf{C}}); \\ \text{com}_T \leftarrow \text{PCOM.CommitPt}(\text{pp}, t, d; r_T); \\ c_y = \text{Eval}(PK, \mathbf{C}, t; r_{c_y}); \\ \text{out}_2(C(\mathbf{C}, t, r_{\mathbf{C}}, r_{c_y}, r_T), R) \\ (\text{pp}, PK, \text{com}_{\mathbf{C}}, \text{com}_T, c_y) = 1 \end{array} \right] = 1$$

**Binding:** For all PPT adversaries  $\mathcal{A}$ , there exists a negligible function  $\epsilon(\cdot)$  such that:

$$\begin{aligned}
& \Pr \left[ \text{pp} \leftarrow \text{PCOM.Setup}(1^\kappa, d); \right. \\
& \quad PK \leftarrow \text{E.KeyGen}(1^\kappa); \\
& \quad (\mathbf{C}_0, r_{\mathbf{C}_0}, \mathbf{C}_1, r_{\mathbf{C}_1}, t_0, r_{T_0}, t_1, r_{T_1}) \leftarrow \mathcal{A}(1^\kappa, n, \text{pp}, PK; r_{\mathcal{A}}); \\
& \quad \text{com}_{\mathbf{C}_0} = \text{PCOM.Commit}(\text{pp}, \mathbf{C}_0; r_{\mathbf{C}_0}) \\
& \quad \text{com}_{\mathbf{C}_1} = \text{PCOM.Commit}(\text{pp}, \mathbf{C}_1; r_{\mathbf{C}_1}) \\
& \quad \text{com}_{T_0} = \text{PCOM.CommitPt}(\text{pp}, t_0, d; r_{T_0}) \\
& \quad \text{com}_{T_1} = \text{PCOM.CommitPt}(\text{pp}, t_1, d; r_{T_1}) \\
& \quad (\text{com}_{\mathbf{C}_0} = \text{com}_{\mathbf{C}_1} \wedge \mathbf{C}_0 \neq \mathbf{C}_1) \\
& \quad \left. \vee (\text{com}_{T_0} = \text{com}_{T_1} \wedge t_0 \neq t_1) \right] \leq \epsilon(\kappa)
\end{aligned}$$

**Witness-Extended Emulation:** For all PPT adversaries  $\mathcal{A}$ , there exists an expected polynomial time emulator  $\mathcal{E}$  and negligible function  $\epsilon(\cdot)$  such that:

$$\begin{aligned}
& \Pr \left[ \text{pp} \leftarrow \text{PCOM.Setup}(1^\kappa, d); \right. \\
& \quad PK \leftarrow \text{E.KeyGen}(1^\kappa); \\
& \quad (\text{com}_{\mathbf{C}}, \text{com}_T, c_y) \leftarrow \mathcal{A}(1^\kappa, n, \text{pp}, PK; r_{\mathcal{A}}); \\
& \quad e \leftarrow (\mathcal{A}(r_{\mathcal{A}}), \text{R})(\text{pp}, PK, \text{com}_{\mathbf{C}}, \text{com}_T, c_y); \\
& \quad (\mathbf{C}, t, r_{\mathbf{C}}, r_{c_y}, r_T) \leftarrow \mathcal{E}^{\mathcal{A}(\text{pp}, PK, \text{com}_{\mathbf{C}}, \text{com}_T, c_y; r_{\mathcal{A}})} \\
& \quad \quad (\text{pp}, PK, \text{com}_{\mathbf{C}}, \text{com}_T, c_y, e) : \\
& \quad (\text{out}_2(e) = 1) \Rightarrow \\
& \quad \quad (\text{com}_{\mathbf{C}} = \text{PCOM.Commit}(\text{pp}, \mathbf{C}; r_{\mathbf{C}}) \\
& \quad \quad \quad \wedge \text{com}_T = \text{PCOM.CommitPt}(\text{pp}, t, d; r_T) \\
& \quad \quad \quad \wedge c_y = \text{Eval}_{PK}(\mathbf{C}, t; r_{c_y})) \left. \right] \geq 1 - \epsilon(\kappa)
\end{aligned}$$

**Honest Verifier Privacy:** There exists a tuple of expected PPT algorithms  $\mathcal{S}$ , given any vector of coefficient of polynomial  $(p_0, \dots, p_d)$  and an evaluation point  $t$ , such that the following distributions are indistinguishable:

$$\begin{aligned}
& - \left\{ \begin{array}{l} \text{pp} \leftarrow \text{PCOM.Setup}(1^\kappa, d); \\ PK \leftarrow \text{E.KeyGen}(1^\kappa); \\ \mathbf{C} \leftarrow (c_0, \dots, c_d) = (\text{Enc}_{PK}(p_0; r_0), \dots, \text{Enc}_{PK}(p_d; r_d)) : \\ \text{com}_{\mathbf{C}} \leftarrow \text{PCOM.Commit}(\text{pp}, \mathbf{C}; r_{\mathbf{C}}); \\ \text{com}_T \leftarrow \text{PCOM.CommitPt}(\text{pp}, t, d; r_t); \\ c_y \leftarrow \text{Eval}_{PK}(\mathbf{C}, t; r_{c_y}); \\ e \leftarrow (\mathbf{C}, t, r_{\mathbf{C}}, r_{c_y}, r_T, r_{\mathcal{A}}), \text{R} \\ (\text{pp}, PK, \text{com}_{\mathbf{C}}, \text{com}_T, c_y) : \\ \text{view}_2(e) \end{array} \right\} \\
& - \left\{ \begin{array}{l} \text{pp} \leftarrow \text{PCOM.Setup}(1^\kappa, d); \\ PK \leftarrow \text{E.KeyGen}(1^\kappa); \\ \mathcal{S}(\text{pp}, PK, d; r_{\mathcal{S}}) \end{array} \right\}
\end{aligned}$$

## 2.2 Our Protocols

In this section, we present the construction of our private polynomial commitment. Our construction is based on the additive homomorphic encryption (AHE) scheme from [13] and the inner-pairing product argument from [16]. As a warm-up, we start by considering a single point where the idea is that the evaluation of a polynomial  $f(x) = \sum_{i=0}^d a_i x^i$  at point  $t$  can be viewed as the inner product between the coefficients vector  $(a_0, a_1, \dots, a_d)$  and the evaluation vector  $T = (1, t, t^2, \dots, t^d)$ . Therefore, given the ciphertexts encrypting the coefficients and the commitments of the evaluation vector  $T$ , the committer proves in Phase 1 that the polynomial evaluation on the ciphertext is indeed the inner product between the two vectors using the techniques in [16]. Next, it remains to show that the committed evaluation vector is well-formed, i.e., it is indeed the powers of the evaluation point  $t$ . To prove this property, denoting the  $i$ -th element in a vector  $T$  as  $T[i]$ , it suffices to show that (1) the 0-th element  $T[0]$  is 1; (2)  $T[i + 1] = T[i] \cdot T[1]$  for  $i = 0, \dots, d - 1$ . These two conditions can further be translated into two types of constraints: linear constraints and quadratic constraints. The first condition is equivalent to the inner product between  $T$  and a public vector  $(1, 0, \dots, 0)$  is 1. For the second condition, we define three selector matrices  $A, B, C \in \mathbb{F}^{d \times (d+1)}$  such that

$$\begin{aligned} X &= A \times T = (T[0], T[1], \dots, T[d - 1]), \\ Y &= B \times T = (T[1], T[1], \dots, T[1]), \\ Z &= C \times T = (T[1], T[2], \dots, T[d]). \end{aligned} \tag{1}$$

Finally, the committer proves that  $X \odot Y = Z$ , where  $\odot$  denotes the Hadamard (element-wise) product. It is not hard to see that  $T$  is the correct evaluation vector if and only if it satisfies these constraints.

We use standard techniques such as [14] to reduce the linear constraints and the quadratic constraints to inner product arguments in Phases 2 and 3. Note that the protocols in these two phases are independent of the ciphertexts encrypting the coefficients. The formal protocol of our private polynomial commitment is presented in Fig. 1. This protocol uses the encryption scheme from [13], the pairing-based commitment from [4] and the Pedersen commitment [52] as building blocks. The protocol also involves private inner product argument, linear constraints test and quadratic constraints test, as described above in the three phases. We present these protocols later in Figs. 3, 4 and 5 together with our scheme for multiple evaluations.

**Multiple Evaluations.** The major advantage of our construction is that it supports batched evaluations on multiple points efficiently, where the proof size and the receiver’s time do not increase by much compared to a single evaluation. We describe our scheme for multiple evaluations in Figs. 2. The differences from the single evaluation variant are highlighted in purple. In particular, in Phase 1 (Steps 1 and 2 in Fig. 2), C and R check the inner products between the coefficient vector in the ciphertext and all the evaluation vectors in the commitments using a single private inner product argument protocol via a random linear combination.



**Setup**( $1^\kappa, d$ ): Generate the public parameters of the bilinear map and the commitment scheme Ped and AFG.  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, e, w, g) \leftarrow \mathcal{G}(1^\kappa)$ ,  $\text{ck}_1 = (w_r, w_0, \dots, w_d), a, b) \leftarrow \text{AFG.KeyGen}(S, 3d + 8)$ ,  $\text{ck}_2 = (v_r, v_0, \dots, v_d) \leftarrow \text{Ped.KeyGen}(1^\kappa, d + 2)$ ,  $\text{ck}_3 \leftarrow \text{Ped.KeyGen}(1^\kappa, 2)$ ,  $\text{ck}_4 = (x_r, x_0, \dots, x_d) \leftarrow \text{Ped.KeyGen}(1^\kappa, d + 2)$ . Output  $pp = (\text{ck}_1, \text{ck}_2, \text{ck}_3, \text{ck}_4, a, b)$ .

**Commit**( $pp, \mathbf{C}, r_C$ ): Given the ciphertext of the coefficients  $\mathbf{C} = (c_0, \dots, c_d)$ , output  $\text{AFG.Commit}_{\text{ck}_1}(\mathbf{C}, g^{r_C}) = e(g^{r_C}, w_r) \cdot \prod_{i=0}^d e(c_i, w_i)$ , where  $r_C \in \mathbb{Z}_p$ .

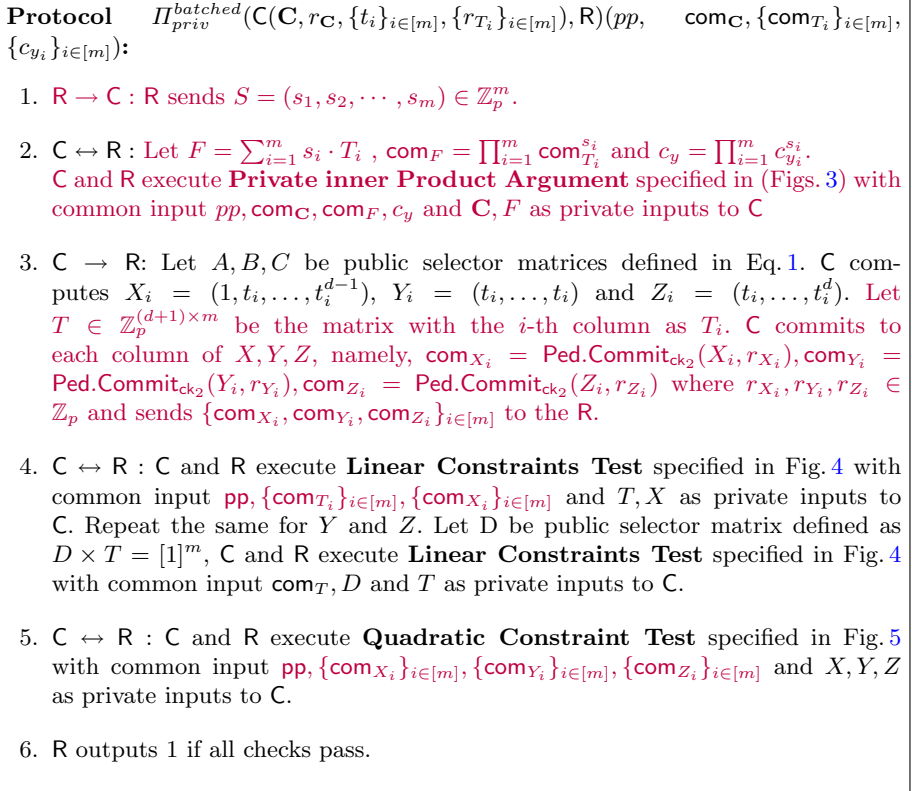
**CommitPt**( $pp, t, r_T, d$ ): Given an evaluation point  $t$ , generate  $T = (1, t, \dots, t^d)$  and output  $\text{Ped.Commit}_{\text{ck}_2}(T, r_T)$  where  $r_T \in \mathbb{Z}_p$ .

**Protocol**  $\Pi_{\text{priv}}(\mathbf{C}(\mathbf{C}, r_C, t, r_T), \mathbf{R})(pp, \text{com}_{\mathbf{C}}, \text{com}_T, c_y)$ :

1. C and R execute **Private inner Product Argument** specified in (Figs. 3) with common input  $pp, \text{com}_{\mathbf{C}}, \text{com}_T, c_y$  and  $\mathbf{C}, T$  as private inputs to C.
2. C  $\rightarrow$  R: Let  $A, B, C$  be public selector matrices defined in Eq. 1. C computes  $X = A \times T = (1, t, \dots, t^{d-1})$ ,  $Y = B \times T = (t, \dots, t)$ ,  $Z = C \times T = (t, \dots, t^d)$ . C commits to  $X, Y, Z$  by  $\text{com}_X = \text{Ped.Commit}_{\text{ck}_2}(X, r_X)$ ,  $\text{com}_Y = \text{Ped.Commit}_{\text{ck}_2}(Y, r_Y)$ ,  $\text{com}_Z = \text{Ped.Commit}_{\text{ck}_2}(Z, r_Z)$ , where  $r_X, r_Y, r_Z \in \mathbb{Z}_p$ . C sends  $\text{com}_X, \text{com}_Y, \text{com}_Z$  to R.
3. C  $\leftrightarrow$  R : C and R execute **Linear Constraints Test** specified in Fig. 4 with common input  $\text{com}_T, \text{com}_X$  and  $T, X$  as private inputs to C. Repeat the same for  $Y$  and  $Z$ . Let  $D$  be public selector matrix defined as  $D \times T = [1]$ , C and R execute **Linear Constraints Test** specified in Fig. 4 with common input  $\text{com}_T, D$  and  $T$  as private inputs to C.
4. C  $\leftrightarrow$  R : C and R execute **Quadratic Constraint Test** specified in Fig. 5 with common input  $\text{com}_X, \text{com}_Y, \text{com}_Z$  and  $X, Y, Z$  as private inputs to C.
5. R outputs 1 if all checks pass.

**Fig. 1.** Private Polynomial Commitments (Single Evaluation).

In Phase 2 (Step 4 in Fig. 2), the product between a selector matrix (i.e.,  $A, B$  or  $C$ ) and all the evaluation vectors can be reduced to a single inner product via two random linear combinations, as shown in Fig. 4. In Phase 3 (Step 5 in Fig. 2), the protocol of the quadratic constraint test is more complicated. We are not able to reduce the Hadamard product of matrices  $X \odot Y = Z$  to a single inner product. Instead, we reduce the Hadamard product to the sum of  $m$  inner products via a random linear combination in Step 1 of Fig. 5. Then we propose a protocol (Step 3 of Fig. 5) to prove the sum of the inner products with a proof size of only  $O(\log d)$ . The protocol is an extension of the scheme for the Hadamard product in [14] in a non-black-box way.



**Fig. 2.** Batched proof for Private Polynomial Commitments. (Color figure online)

**Theorem 1.** *Protocol PCOM (Fig. 2) is a private polynomial commitment scheme as in Definition 1, under the Decisional Linear (DLIN) and the Double Pairing Problem (DPP) hardness assumptions.*

**Proof Sketch:** To show PCOM is a private polynomial commitment scheme (Definition 1), we show that the protocol satisfies completeness, binding, witness-extended emulation and honest verifier privacy.

**Completeness:** In the private inner product argument test, there are two phases - the masking phase and the inner product phase. In the end,  $\mathbf{R}$  accepts if the combined commitment of the private polynomial, evaluation vector and evaluation ciphertext is decommitted correctly. This essentially follows from showing that the commitment of the private polynomial, the commitment of evaluation vector and evaluation ciphertext are updated correctly in each round. The rest of the protocol involving the linear constraint test, quadratic test and the BBB-IPA follow essentially observing that the corresponding constraints are satisfied.

**Private inner Product Argument**

Private Inputs:  $\mathbf{C}$ :  $\mathbf{C} = (c_0, \dots, c_d) \in \mathbb{G}_E^{d+1}, F = (f_0, \dots, f_d) \in \mathbb{Z}_p^{d+1}$ .

Public Inputs:  $pp = (\text{ck}_1, \text{ck}_2, \text{ck}_3, a, b, \text{PK}), \text{com}_{\mathbf{C}}, \text{com}_F, c_y$ .

**1. Masking Phase:**

- (a)  $\mathbf{C} \rightarrow \mathbf{R}$ :  $\mathbf{C}$  generates a random encrypted polynomial  $\mathbf{E} = (e_0, \dots, e_d) \in \mathbb{G}_E^{d+1}$  where  $e_i = \text{Eval}_{\text{PK}}(r_i)$  and  $r_i \in \mathbb{Z}_p$ . A random vector  $M = (M_0, \dots, M_d) \in \mathbb{Z}_p^{d+1}$  is also sampled and generates commitment  $\text{com}_{\mathbf{E}} = \text{AFG.Commit}_{\text{ck}_1}(\mathbf{E}, r_{\mathbf{E}})$  and  $\text{com}_M = \text{Ped.Commit}_{\text{ck}_2}(M, r_M)$  where  $r_{\mathbf{E}}, r_M \in \mathbb{Z}_p$ .  $\mathbf{C}$  also computes:  $c_l = \langle \mathbf{E}, F \rangle$ ,  $c_r = \langle \mathbf{C}, M \rangle$ ,  $c_m = \langle \mathbf{E}, M \rangle$  and sends  $\text{com}_{\mathbf{E}}, \text{com}_M, c_l, c_r, c_m$  to  $\mathbf{R}$ .
- (b)  $\mathbf{R} \rightarrow \mathbf{C}$ :  $\mathbf{R}$  sends a random challenge  $x \in \mathbb{Z}_p$ .
- (c) Both parties set  $\text{com}'$  where:  $\text{com} = \text{com}_{\mathbf{C}} \cdot e(\text{com}_F, a) \cdot e(c_y, b)$ ,  $\text{com}' = \text{com} \cdot \text{com}_{\mathbf{E}}^x \cdot e(\text{com}_M, a)^{x^{-1}} \cdot e(c_l^x \cdot c_m \cdot c_r^{x^{-1}}, b)$ , and  $\mathbf{C}$  sets  $\mathbf{C}' = \mathbf{C} \odot \mathbf{E}^x$  and  $F' = F + x^{-1} \cdot M$  where  $\odot$  denotes element-wise multiplication of two vectors.
- (d) Both parties update  $\text{com} = \text{com}'$ ,  $\mathbf{C} = \mathbf{C}'$ ,  $F = F'$ .

**2. Inner Product Phase:** For round  $\text{rnd} = 1$  to  $\log d - 1$ :

- (a) Set  $d' = (d + 1)/2$ .  $\mathbf{C}$  sets  $\mathbf{C}_L = \mathbf{C}[d']$ ,  $\mathbf{C}_R = \mathbf{C}[d' : ]$ ,  $F_L = F[d']$  and  $F_R = F[d' : ]$  while both  $\mathbf{C}$  and  $\mathbf{R}$  sets  $\text{ck}_{1L} = \text{ck}_1[d']$ ,  $\text{ck}_{1R} = \text{ck}_1[d' : ]$ ,  $\text{ck}_{2L} = \text{ck}_2[d']$ , and  $\text{ck}_{2R} = \text{ck}_2[d' : ]$ .
- (b)  $\mathbf{C}$  generates intermediate cross-commitments:  $\text{com}_{\mathbf{C}_L} = \text{AFG.Commit}_{\text{ck}_{1R}}(\mathbf{C}_L, r_{\mathbf{C}_L})$ ,  $\text{com}_{\mathbf{C}_R} = \text{AFG.Commit}_{\text{ck}_{1L}}(\mathbf{C}_R, r_{\mathbf{C}_R})$ ,  $\text{com}_{F_L} = \text{Ped.Commit}_{\text{ck}_{2R}}(F_L, r_{F_L})$ ,  $\text{com}_{F_R} = \text{Ped.Commit}_{\text{ck}_{2L}}(F_R, r_{F_R})$ , where  $r_{\mathbf{C}_L}, r_{\mathbf{C}_R}, r_{F_L}, r_{F_R} \in \mathbb{Z}_p$ .
- (c)  $\mathbf{C} \rightarrow \mathbf{R}$ :  $\mathbf{C}$  generated  $L$  and  $R$ :  $c_l = \langle \mathbf{C}_R, F_L \rangle$ ,  $c_r = \langle \mathbf{C}_L, F_R \rangle$   
 $L = \text{com}_{\mathbf{C}_R} \cdot e(\text{com}_{F_L}, a) \cdot e(c_l, b)$ ,  $R = \text{com}_{\mathbf{C}_L} \cdot e(\text{com}_{F_R}, a) \cdot e(c_r, b)$ , where  $a, b \in pp$  and sends  $L, R$  to  $\mathbf{C}$ .
- (d)  $\mathbf{R} \rightarrow \mathbf{C}$ :  $\mathbf{R}$  sends a random challenge  $x \in \mathbb{Z}_p$ .
- (e)  $\mathbf{C}$  sets  $\mathbf{C}' = \mathbf{C}_L \odot \mathbf{C}_R^x$  and  $F' = F_L + x^{-1} \cdot F_R$  where  $\odot$  denotes element-wise multiplication of two vectors while  $\mathbf{C}$  and  $\mathbf{R}$  both locally compute the new keys  $\text{ck}'_1 = \text{ck}_{1L} \odot \text{ck}_{1R}^{x^{-1}}$  and  $\text{ck}'_2 = \text{ck}_{2L} \odot \text{ck}_{2R}^x$ .
- (f)  $\mathbf{R}$  computes new commitment  $\text{com}' = L^x \cdot \text{com} \cdot R^{x^{-1}}$ .
- (g)  $\mathbf{C}$  and  $\mathbf{R}$  will update  $\mathbf{C} = \mathbf{C}'$ ,  $F = F'$ ,  $\text{com} = \text{com}'$ , and  $\text{ck}_i = \text{ck}'_i \forall i \in [2]$   
 In round  $\log d$ :
- (h) In the last round,  $\mathbf{C}$  opens  $\text{com}$  to  $\mathbf{C}'$ ,  $F'$  and  $c'_y$  and  $\mathbf{R}$  accepts if  $c_y = \langle \mathbf{C}', F' \rangle$ .
- (i) If all checks pass,  $\mathbf{R}$  outputs  $b = 1$  else output  $b = 0$ .

**Fig. 3.** Private Inner Product Argument.

**Binding:** To argue the binding property of PCOM, it can be trivially reduced to the binding property of the Ped and AFG commitment scheme.

**Witness-Extended Emulation:** To argue witness-extended emulation of PCOM, as shown in [14], it is enough to show that given  $(n_1, \dots, n_r)$ -tree of

**Linear constraint Test (Prove  $A \times T = X$ )**

- Private Inputs: C has private inputs:  $X \in \mathbb{Z}_p^{d \times m}, T \in \mathbb{Z}_p^{d+1 \times m}$ .
- Public Inputs:  $pp = (\text{ck}_1, \text{ck}_2, \text{ck}_3, a, b, \text{PK}), \{\text{com}_{T_i}\}_{i \in [m]}, \{\text{com}_{X_i}\}_{i \in [m]}$  where  $\text{com}_{T_i}, \text{com}_{X_i} \in \mathbb{G}_1$ .
  1.  $R \rightarrow C$ : R sends random vectors  $S \in \mathbb{Z}_p^d$  and  $U \in \mathbb{Z}_p^m$ . Let  $S_A = S \times A$ ,  $T_U = T \times U$ ,  $X_U = X \times U$ . We observe that if  $A \times T = X$  then for any  $S \in \mathbb{Z}_p^d$  and  $U \in \mathbb{Z}_p^m$  we have:
 
$$S \times A \times T \times U = S \times X \times U, \text{ i.e., } \langle S_A, T_U \rangle - \langle S, X_U \rangle = 0.$$
  2.  $C \rightarrow R$ : C computes two cross terms inner product  $l$  and  $r$  and sends their respective commitments  $\text{com}_l, \text{com}_r$  to R:  $l = \langle S_A, -X_U \rangle, r = \langle S, T_U \rangle$ ,  $\text{com}_l = \text{Ped.Commit}_{\text{ck}_3}(l, r_l), \text{com}_r = \text{Ped.Commit}_{\text{ck}_3}(r, r_r)$ , where  $r_l, r_r \in \mathbb{Z}_p$ .
  3.  $R \rightarrow C$ : R sends a random challenge  $x \in \mathbb{Z}_p$ .
  4.  $R \leftrightarrow C$ : C computes  $L = S_A + x^{-1} \cdot S$  and  $R = T_U - x \cdot X_U$ . C and R both compute  $\text{com}_L = \text{Ped.Commit}_{\text{ck}_4}(S_A + x^{-1} \cdot S; 0)$  and  $\text{com}_R = \prod_{i=1}^m \text{com}_{T_i}^{U^{[i-1]}} \cdot \text{com}_{X_i}^{U^{[i-1]} \cdot x}$ . C and R execute BBB-IPA on common inputs is  $\text{ck}_4, \text{ck}_2, \text{ck}_3, \text{com}_L, \text{com}_R, \text{com}_l^x \cdot \text{com}_r^{x^{-1}}$  and private inputs of C are  $L, R, x \cdot l + x^{-1} \cdot r$ .
  5. If all checks pass, R outputs  $b = 1$  else output  $b = 0$ .
- A special case is when  $D \times T = X$  where  $X$  is a known vector of dimensions  $1 \times m$ . The above test can be simplified where R sends a random vector  $U \in \mathbb{Z}_p^m$  and the check is reduced from  $D \times T = X$  to  $\langle D, T_U \rangle = d$  where  $T_U = T \times U$  and  $d = \sum_{i=0}^{m-1} U[i]$ . C and R compute  $\text{com}_D = \text{Ped.Commit}_{\text{ck}_4}(D, 0)$ ,  $\text{com}_{T_U} = \prod_{i=1}^m \text{com}_{T_i}^{U^{[i-1]}}$ ,  $\text{com}_d = \text{Ped.Commit}(d, 0)$ . C and R execute BBB-IPA on common inputs is  $\text{ck}_4, \text{ck}_2, \text{ck}_3, \text{com}_D, \text{com}_{T_U}, \text{com}_d$  and private inputs of C are  $D, T_U, d$ . If all checks pass, R outputs  $b = 1$  else output  $b = 0$ .

**Fig. 4.** Linear Constraint Test.

accepting transcripts, there exist a PPT extractor  $\mathcal{X}$  which extracts the witness for PCOM. To construct  $\mathcal{X}$ , we first construct a witness-extraction algorithm  $\mathcal{X}_1$  that succeeds in extracting the witness of Private Inner Product Argument given  $(n_1, \dots, n_r)$ -tree of accepting transcripts. Using the rewinding property of the extractor and choosing different randomness in each rewinding, the extractor  $\mathcal{X}_1$  can extract the witness. Here, the witness is the encrypted polynomial, evaluation vector, encrypted evaluation and the randomness used to generate the commitments. Next  $\mathcal{X}$  extracts the evaluation vector from Linear Test and Quadratic test to verify if the evaluation used in all three tests is the same. We use the witness-extended emulation extractor of BBB-IPA as a subprotocol in extracting the evaluation vector from the Linear and Quadratic tests.

**Honest Verifier Privacy:** To show honest verifier privacy, we construct a simulator  $\mathcal{S}$ . Indistinguishability of the simulation essentially follows from semantic security of the underlying encryption scheme, hiding of the commitment scheme,

**Quadratic Constraint Test (Prove  $X \odot Y = Z$ )**

Private Inputs:  $C : X, Y, Z \in \mathbb{Z}_p^{d \times m}$ .

Public Inputs:  $pp = (\text{ck}_1, \text{ck}_2, \text{ck}_3, \text{ck}_4, a, b, \text{PK}), \{\text{com}_{X_i}\}_{i \in [m]}, \{\text{com}_{Y_i}\}_{i \in [m]}, \{\text{com}_{Z_i}\}_{i \in [m]}$  where  $\text{com}_{X_i}, \text{com}_{Y_i}, \text{com}_{Z_i} \in \mathbb{G}_1$ .

1.  $R \rightarrow C$ :  $R$  sends a random vector  $S \in \mathbb{Z}_p^m$  and a random value  $w$ . Now if  $X \odot Y = Z$ , then  $\sum_{i \in m} w^i (\langle X_i, Y_i \odot S \rangle - \langle Z_i, S \rangle) = 0$ .
2. Let  $L_i = w^i \cdot X_i, L_{i+m} = w^i \cdot Z_i, R_i = Y_i \odot S, R_{i+m} = -S$   $C$  and  $R$  compute a new key  $\text{ck}_5$  where  $\text{ck}_5[j] = \text{ck}_2^{S[j]-1}[j]$  for all  $j \in [0, d]$  and compute the commitments as follows:  $\text{com}_{L_i} = \text{com}_{X_i}^{w^i}, \text{com}_{L_{i+m}} = \text{com}_{Z_i}^{w^i}, \text{com}_{R_i} = \text{com}_{Y_i}, \text{com}_{R_{i+m}} = \text{Ped.Commit}_{\text{ck}_5}(-S)$
3.  $C$  sets  $d = 0$  while  $R$  sets  $\text{com}_d = 1$ . Also set  $m' = 2m$ .

For round 1 to  $\log m$ :

- (a)  $C \rightarrow R$ : Set  $m' = m'/2$ .  $C$  computes two cross terms inner product  $l = \sum_{i=1}^{m'} \langle L_i, R_{i+m'} \rangle$  and  $r = \sum_{i=1}^{m'} \langle L_{i+m'}, R_i \rangle$  and sends a  $\text{Ped}$  commitment of these two ( $\text{com}_l$  and  $\text{com}_r$ ) to  $R$ .  
where  $r_l, r_r \in \mathbb{Z}_p$ .
- (b)  $R \rightarrow C$ :  $R$  sends a random challenge  $x \in \mathbb{Z}_p$ .
- (c)  $C$  computes  $\{L'_i = L_i + x^{-1} \cdot L_{i+m}\}_{i \in [m']}$  and  $\{R'_i = R_i + x \cdot R_{i+m}\}_{i \in [m']}$  while  $R$  updates the commitments  $\text{com}_{L'_i} = \text{com}_{L_i} \cdot \text{com}_{L_{i+m}}^{x^{-1}}$  and  $\text{com}_{R'_i} = \text{com}_{R_i} \cdot \text{com}_{R_{i+m}}^x$ .
- (d)  $C$  computes  $d' = d + x \cdot l + x^{-1} \cdot r$  while  $R$  computes  $\text{com}_{d'} = \text{com}_d \cdot \text{com}_l^x \cdot \text{com}_r^{x^{-1}}$ .
- (e)  $C$  updates  $L_i = L'_i, R_i = R'_i, d = d'$  while  $R$  updates  $\text{com}_d = \text{com}_{d'}$ .

In round  $\log m + 1$ :

- (f)  $C$  sets  $L = L_1$  and  $R = R_1$  while  $R$  sets  $\text{com}_L = \text{com}_{L_1}$  and  $\text{com}_R = \text{com}_{R_1}$   $C$  and  $R$  execute BBB-IPA on instance with common input  $\text{ck}_2, \text{ck}_5, \text{ck}_3, \text{com}_L, \text{com}_R, \text{com}_d$  and  $L, R, d$  as private inputs of  $C$ .

**Fig. 5.** Quadratic Constraint Test.

honest-verifier zero-knowledge property of the underlying BBB-IPA and standard masking techniques.

**Complexity.** The communication complexity of our polynomial commitments is  $O(\log d)$  for a single evaluation and  $O(m + \log d)$  for  $m$  points where  $d$  is the degree of the polynomial. Their round complexity is  $O(\log m + \log d)$  rounds.

The computational complexity of the committer is  $O(m \cdot d)$  modular exponentiations and  $O(d)$  bilinear pairings, while the complexity of the receiver is  $O(m + d)$  exponentiations. The space complexity of our private polynomial commitment scheme is  $O(m + d)$  for the committer as it needs to store the encrypted polynomial and the evaluation points. The space complexity of the receiver is  $O(m)$  (resp.  $O(m + \log d)$ ) in the interactive (resp. non-interactive setting). This difference is due to the fact that in the non-interactive setting, the entire proof is stored for validation.

### 3 Scalable Multi-party PSI

Our first application is a new scalable PSI protocol that follows the blueprint of [43]. This protocol is carried out in a star topology network with  $P_1$  being the central party. In this work, we show that the actions of  $P_1$  can be captured by the abstraction of a private polynomial commitment.

We broadly split our protocol description into four main phases. In the first phase (Key Generation), the parties jointly generate a public key without disclosing their corresponding secret key shares, as well as the public parameters for the two polynomial commitments. The second phase (Commitment Phase) is executed by the central party  $P_1$  that broadcasts commitments of its input together with a proof of knowledge. In the third phase (Aggregation), all parties (except  $P_1$ ) send it an encrypted polynomial whose roots correspond to their inputs.  $P_1$  combines these polynomials for each party and provides a commitment of the encrypted aggregated polynomial while proving the correctness of aggregation. The last phase (Intersection) concludes the protocol by extracting the intersection, where  $P_1$  evaluates the aggregated polynomial on its input and provides proof of correct evaluation. Once the proof is validated, the parties decrypt each evaluation to get the intersection.

Our polynomial commitments will be useful in [43] for two purposes; proving the correctness of aggregation by evaluating on a public point and proving the correctness of evaluations on  $P_1$ 's input finally to reveal the intersection.

We use the following primitives in our construction:

- A threshold additively homomorphic encryption scheme with protocols ( $\Pi_{\text{GEN}}$  and  $\Pi_{\text{DecZero}}$ ) to respectively sample a public-key together with the secret key shares, and a protocol to determine if a target ciphertext decrypts to 0. We instantiate our scheme with BBS encryption scheme [13] which relies on DLIN assumption.
- Our polynomial commitment scheme PCOM, (that is compatible with the threshold encryption scheme), and is instantiated with non-interactive publicly verifiable proofs of evaluation of hidden points (in the batched setting) and public points (in the single instance setting). We respectively denote the committer and receiver algorithms for the corresponding (non-interactive) proof systems by  $(\text{PCOM.C}_{\text{hid}}^{\text{batch}}, \text{PCOM.R}_{\text{hid}}^{\text{batch}})$  and  $(\text{PCOM.C}_{\text{pub}}, \text{PCOM.R}_{\text{pub}})$ . To construct PCOM, we require two commitment schemes: Pederson Commitment scheme [52] which relies on the DL assumption and the AFG Commitment scheme [4] which is based on bilinear pairing and relies on the DPP assumption.
- An  $n$ -party protocol  $\Pi_{\text{COIN}}$  to sample random coins.
- A simulation extractable non-interactive publicly verifiable proof system  $\Pi_{\text{EXP}}$  to prove knowledge of exponent. We instantiate this with the non-interactive variant of the classic protocol due to [58] via the Fiat-Shamir transform. We denote the prover and verifier algorithms by  $(\text{DL.P}_{\text{pub}}, \text{DL.V}_{\text{pub}})$ .

**Protocol  $\pi_{\text{MPSI}}$  with Malicious Security (Part 1)**

**Input:** Party  $P_i$  is given a set  $X_i = \{x_i^1, \dots, x_i^{m_i}\}$  of size  $m_i$  for all  $i \in [n]$ . All parties are given a security parameter  $1^\kappa$  and a description of a group  $\mathbb{G}$ .

**The protocol:**

1. **Key Generation.** The parties mutually generate a public key PK and the corresponding secret key shares  $(\text{SK}_1, \dots, \text{SK}_n)$  by running  $\pi_{\text{GEN}}$ .  $P_1$  also runs the setup for the polynomial commitment scheme by running  $\text{PCOM.Setup}(1^\kappa, m_{\text{max}})$ .
2. **Commitment phase.**  $P_1$  creates commitments to its inputs  $\{\text{com}_{T_1}, \dots, \text{com}_{T_n}\}$  where  $\text{com}_{T_i} = \text{PCOM.CommitPt}(\text{pp}, x_i^1, r_{T_i}, m_{\text{max}})$  and  $r_{T_i} \in \mathbb{Z}_p$  is randomly chosen and generates a proof using DL.P proving knowledge of the committed message and broadcasts the commitment and proof to all parties.
3. **Aggregation**
  - (a) For all  $i \in [2, n]$ , party  $P_i$  computes the coefficients of a polynomial  $A_i(\cdot) = (a_0^i, \dots, a_{m_i}^i)$  of degree  $m_i$ , with roots set to the  $m_i$  elements of  $X_i$ . In addition,  $P_i$  chooses a random element  $\lambda_i \leftarrow \mathbb{G}$  and computes the product  $\lambda_i \cdot a_j^i$  for every coefficient within  $A_i$ .  $P_i$  sends  $P_1$  the sets of ciphertexts  $\mathbf{C}_i = (c_0^i, \dots, c_{m_i}^i)$ , encrypting the coefficients of  $\lambda_i \cdot A_i(\cdot)$ .
  - (b) Upon receiving the ciphertexts from all parties, party  $P_1$  combines the following ciphertexts

$$c_0 = \prod_{i=2}^n c_0^i, \dots, c_{m_{\text{max}}} = \prod_{i=2}^n c_{m_{\text{max}}}^i$$

where  $m_{\text{max}} = \max(m_2, \dots, m_n)$ . Note that  $P_1$  generates the ciphertexts by encrypting the coefficients of the combined polynomial  $A(\cdot) = \lambda_2 \cdot A_2(\cdot) + \dots + \lambda_n \cdot A_n(\cdot)$ .  $P_1$  then generates and broadcasts  $\text{com}_{\mathbf{C}}$  which is a commitment of the encrypted polynomial  $\mathbf{C}(\cdot) = (c_0, \dots, c_{m_{\text{max}}})$  using  $\text{PCOM.Commit}(\text{pp}, \mathbf{C}, r_{\mathbf{C}})$  where  $r_{\mathbf{C}}$  is generated randomly.

- (c) Next, the parties verify whether the polynomials aggregation was done correctly. Specifically, the parties first agree on a random element  $u$  from the appropriate plaintext domain using the coin tossing protocol  $\pi_{\text{COIN}}$ .  $P_1$  broadcasts the encrypted evaluation  $\tilde{\lambda} = \text{Eval}(\text{PK}, \mathbf{C}, u)$  along with a proof of correct evaluation by using  $\text{PCOM.C}_{\text{pub}}$  on public inputs  $\text{pp}, \text{com}_{\mathbf{C}}, u, \tilde{\lambda}$  and private inputs  $\mathbf{C}, r_{\mathbf{C}}$ .
- (d) Then, each party broadcasts the ciphertext  $\tilde{\lambda}_i = \text{Eval}(\text{PK}, \mathbf{C}_i, u)$ , together with a ZK proof of knowledge generated using DL.P for proving the knowledge of the plaintext. If all the proofs are verified correctly, then the parties check that  $\tilde{\lambda} - \prod_{i=2}^n \tilde{\lambda}_i$  encodes a 0-message using  $\pi_{\text{DecZero}}$ .

**Fig. 6.** Multi-party PSI protocol (Part 1).

The protocol is split into two parts and presented in Figs. 6 and 7. The first three phases of the protocol: Key Generation, Commitment Phase and Aggregation are covered in Fig. 6 whereas the Intersection is contained in Fig. 7.

**Theorem 2.** *The protocol  $\pi_{\text{MPSI}}$  described in Figure 6 and Fig. 7 securely realizes  $\mathcal{F}_{\text{MPSI}}$  in the presence of malicious adversaries and dishonest majority*

**Protocol  $\pi_{\text{MPSI}}$  with Malicious Security (Part 2)**

**The protocol (continued):**

4. **Intersection.**

(a) If the above verification is completed correctly,  $P_1$  evaluates the aggregated polynomial that is encrypted within ciphertexts  $\mathbf{C} = (c_1, \dots, c_{m_{\max}})$ , on its input elements  $\{x_1^j\}_{j=1}^{m_1}$ , and proves consistency with the commitment  $\text{com}_{\mathbf{C}}$ .  $P_1$  forwards the encrypted evaluations  $c_y = \text{Eval}(PK, \mathbf{C}, t)$  along with a proof generated using  $\text{PCOM.C}_{\text{hid}}^{\text{batch}}$  on public inputs  $\text{pp}, \text{com}_{\mathbf{C}}, \{\text{com}_{T_i}\}_{i \in [m]}, \{c_{y_i}\}_{i \in [m_1]}$  and private inputs  $\mathbf{C}, r_{\mathbf{C}}, X_1, \{r_{T_i}\}_{i \in [m_1]}$

(b) All parties verify the evaluations and then decrypt the evaluations using protocol  $\pi_{\text{DecZero}}$  to reveal the intersection.

**Fig. 7.** Multi-party PSI protocol (Part 2).

**Table 3.** MPSI Communication Complexity.

	$P_1$	$P_i$	Total
KeyGen	$O(n)$	$O(n)$	$O(n^2)$
Commit	$O(n \cdot m_{\min})$	—	$O(n \cdot m_{\min})$
Aggregate	$O(n \cdot \log m_{\max})$	$O(m_i + n)$	$O(n^2 + \sum_{i=2}^n m_i + n \cdot \log m_{\max})$
Intersection	$O(n \cdot (m_{\min} + \log m_{\max}))$	$O(m_{\min})$	$O(n \cdot (m_{\min} + \log m_{\max}))$
MPSI	$O(n \cdot (m_{\min} + \log m_{\max}))$	$O(n + m_{\min} + m_i)$	$O(n^2 + \sum_{i=1}^n m_i + n \cdot (m_{\min} + \log m_{\max}))$

under Decisional Linear (DLIN) and Double Pairing Problem (DPP) hardness assumptions.

**Proof Sketch:** We split the analysis into two cases based on whether the set of corrupted parties includes the central party  $P_1$  or not. Consider an adversary  $\mathcal{A}$  that corrupts a set of parties that includes  $P_1$ . We define a simulator  $\mathcal{S}$  and prove that the real and simulated executions are computationally indistinguishable. The indistinguishability between the real and simulated execution is reduced to the privacy property of the encryption scheme, the hiding property of the commitment schemes, and the privacy property of the polynomial commitment. In the first case, the central party  $P_1$  is corrupted, and the input of  $P_1$  can be extracted from  $P_1$ 's input commitment in the commit phase. The input of other corrupted parties can be extracted by rewinding the aggregation phase. This is achieved by extracting  $d + 1$  evaluation points of every corrupted party's polynomial as shown in [43]. In the second case, the simulation is the same as the previous case with the exception that it does not need to extract  $P_1$ 's input.



**Complexity.** The communication complexity of our protocol is linear in the input sizes and the number of parties, where the smallest input size can be given to  $P_1$ . Naively, the communication complexity of our protocol is  $O(n^2 + \sum_{i=1}^n m_i + n \cdot m_{\min} \cdot \log m_{\max})$  when the polynomial commitment is separately used for each evaluation point. The batching feature of our scheme reduces the communication cost of our protocol to  $O(n^2 + \sum_{i=1}^n m_i + n \cdot (m_{\min} + \log m_{\max}))$ . For the central party  $P_1$ , the communication cost is  $O(n(m_{\min} + \log m_{\max}))$ .  $P_1$  generates a batched evaluation proof of size  $O(m_{\min} + \log m)$ . The dominating cost for  $P_1$  is sending the evaluation proof to all other parties. For all other parties, the communication cost is  $O(n + m_{\min} + m_i)$  where  $O(n)$  is sent during the Key Generation phase as well as verifying the aggregation. Additionally, the communication cost in sending the encrypted polynomial to  $P_1$  and generating the intersection is  $O(m_i)$  and  $O(m_{\min})$  respectively. We provide a detailed analysis in Table 3, providing the communication complexity of the parties individually as well as together along every phase of the MPSI protocol. The round complexity of our protocol is dominated by the round complexity of the underlying polynomial commitments. In the random oracle model, the round complexity is 4.

Computationally, the dominating part of the protocol is evaluating the aggregated polynomial and executing the private polynomial commitment from Sect. 2. The complexity of our protocol is  $O(m_{\max} \cdot m_{\min})$  exponentiations. We further reduce the polynomial degrees and the overall workload using hashing techniques; see below for more details. The space complexity of our protocol in the interactive setting is  $O(m_{\max})$  for  $P_1$  and  $O(m_i)$  for every other party  $P_i$ , while in the non-interactive setting the complexity is  $O(m_{\max})$  for  $P_1$  and  $O(m_i + \log m_{\max})$  for party  $P_i$ . We note that the space complexity of  $P_1$  is independent of the number of parties. In particular, the polynomials received by the parties can be aggregated on-the-fly and do not require any extra space. Regarding the polynomial commitments, the non-interactive variant requires  $P_i$  to store the entire proof in the memory which increases the space complexity by an additive factor of  $O(\log m_{\max})$ .

**Hashing.** A notable optimization in PSI protocols is using simple hashing to map the input into smaller sets (buckets), and running a different instance per bucket. In our context, this enables us to reduce the workload of  $P_1$  from quadratic to quasilinear. The idea behind simple hashing lies in splitting the input set into bins where based on a hash function, each element is assigned to a bin. Next, the parties sort their input into bins and run an MPSI protocol separately on each bin. Splitting the input into bins reduces the size of the degree of the polynomials and improves the computation cost of the parties for the computationally heavy tasks of polynomials interpolations and evaluations.

Simple hashing can be directly used in the malicious setting where each bin induces a separated polynomial. Note that the adversary can only attempt to put an item in a wrong bin but this item can be ignored by the simulator. Let  $h$  be a hash function,  $m_{\max}$  be the maximum number of items in an input set,  $\mathcal{B}$  be

the number of bins and  $M$  is the maximum of items in a bin. It is known that if a hash function maps  $m_{\max}$  items into  $\mathcal{B}$  bins and  $m_{\max} \geq \mathcal{B} \log \mathcal{B}$  then with very high probability,  $M = \frac{m_{\max}}{\mathcal{B}} + \sqrt{\frac{m_{\max} \log \mathcal{B}}{\mathcal{B}}}$  [56, 64]. Setting  $\mathcal{B} = \frac{m_{\max} \log \log m_{\max}}{\log m_{\max}}$  and applying the Chernoff bound implies that  $M = O\left(\frac{\log m_{\max}}{\log \log m_{\max}}\right)$  with negligible error in  $m_{\max}$ . Simple hashing can be used to reduce the number of exponentiations, thereby reducing the computational cost. Namely, for each bin, the number of required exponentiations is  $O(M^2)$  and the overall number of exponentiations will be  $O(\mathcal{B}M^2)$ . Substituting the values of  $\mathcal{B}$  and  $M$  using the above analysis will result in  $O\left(m_{\max} \frac{\log m}{\log \log m}\right)$  exponentiations. We refer to Sect. 5 for more details regarding the concrete improvement.

The hashing techniques are not useful for improving [9] as they cannot be broken into small instances. While the improvement for [33] will potentially be smaller since its computational complexity is quasilinear in the input size.

## 4 Other Applications

In this section, we consider a list of distributed tasks in different settings, whose realization can make use of private polynomial commitments. All applications can benefit from the batching of our scheme while achieving malicious security.

### 4.1 Oblivious Polynomial Evaluation

Following the discussion from Sect. 1, in this work, we consider a distributed variant of the oblivious polynomial evaluation functionality denoted by DOPE, where the polynomial  $Q_i(\cdot)$  is linearly shared amongst a set of  $n - 1$  parties. More formally, we define the DOPE functionality as follows. The input of party  $P_i$  for  $i \in [2, n]$  is a polynomial  $Q_i(\cdot)$  of degree at most  $d$  whereas the input of  $P_1$  is an element  $t$ , and the goal is that  $P_1$  learns  $\sum_{i \in [2, n]} Q_i(t)$ .

**Table 4.** Comparison between different DOPE protocols where comm refers to the communication complexity and comp refers to the computational complexity (stated as the number of exponentiations),  $\kappa$  is the computational security parameter,  $\lambda$  is the statistical security parameter,  $n$  is the number of parties and  $d$  is the degree of the polynomial.

	$P_1$ comm	$P_i$ comm	Total comm	$P_1$ comp	$P_i$ comp
[42]	$O(n(d\kappa) + n\lambda)$	$O(d\lambda\kappa)$	$O((n + \lambda)d\kappa)$	$O(nd\lambda)$	$O(d\lambda)$
[40]	$O(n\kappa \log d)$	$O(d\kappa)$	$O(nd\kappa)$	$O(nd)$	$O(d)$
Our Work	$O(n\kappa \log d)$	$O(d\kappa)$	$O(nd\kappa)$	$O(d)$	$O(d)$

We can realize our DOPE functionality in the presence of  $n - 1$  malicious corruptions based on our polynomial commitment scheme following the blueprint

of our PSI protocol. Namely, the parties send their encrypted coefficients to  $P_1$  that aggregates the ciphertexts and evaluates  $Q(\cdot)$  on its input  $t$ .  $P_1$  further attaches proofs of correct aggregation and evaluation. Finally, the parties run a distributed decryption protocol for  $P_1$  to learn  $Q(t)$ . Note that, while in PSI the inputs of the parties are extracted from the polynomials' roots, here the inputs are the polynomial's shares that form  $Q(\cdot)$ .

Our scheme is further flexible regarding the level of threshold introduced by the underlying secret sharing scheme. In particular, one may use any threshold linear secret sharing for splitting the polynomial into shares (rather than simple additive sharing), where the threshold parameter can be smaller than  $n - 1$ . We also have a simple aggregation mechanism which allows the DOPE to be reduced to a single OPE execution where  $n - 1$  parties play the role of  $P_2$ .

Two prior OPE constructions with malicious security [40, 42] can be extended to the distributed setting, where each party  $P_i$  for  $i \in [2, n]$  carries out an individual OPE with  $P_1$ . Compared to previous work; see Table 4, our construction achieves better computational complexity for the central party  $P_1$  due to the fact that the aggregation mechanism allows  $P_1$  to combine the polynomials cheaply and then run the protocol with almost the same cost as running a two-party instance of OPE. The overall communication complexity of our protocol is similar to [40] and is better than [42].

Finally, we note that we can further extend our protocol to support multivariate polynomials to cover a broader class of functionalities.

## 4.2 Verifiable Polynomial Evaluations

In this setting, we focus on verifying the evaluations of a polynomial  $Q(\cdot)$ , linearly shared across a set of  $n - 1$  clients, that are aggregated and stored by a cloud server. Specifically, a set of clients outsource their shares of a  $d$ -degree polynomial (potentially in the clear), to an untrusted server while storing a short state. The server stores the aggregated polynomials and prepares a proof for this computation. Next, whenever the clients provide an input  $x$ , the server computes  $Q(x)$  and a short proof that allows the clients to verify this computation in sub-linear time in  $d$ . We require the verification process to be *public*. Finally, the clients output  $Q(x)$ .

Employing our polynomial commitment by the server, the clients can non-interactively verify the proofs it provides. Furthermore, our solution supports the feature that the polynomial may also be kept private since the shares can be stored on the server while encrypted, where only the evaluation points are public. In more details, each party  $P_i$  sends the server its polynomial share  $Q_i(\cdot)$ . The server aggregates the shares and computes a proof of correct aggregation (that can be made non-interactive by using the random oracle to choose the random evaluated point for this test). Upon receiving an input  $x$ , the parties forward it to the server that computes (the encryption of)  $Q(x)$  together with a proof of correctness. Our protocol is secure in the presence of  $n - 2$  corrupted clients, and a colluding server. Note that the degree of  $Q(\cdot)$  may be huge, yet

uploading it is a one time phase whose complexity amortizes away over multiple evaluation points. Moreover, the proofs of correct evaluations can be batched.

A related modeling is multi-clients verifiable computation where a set of clients wish to compute some function  $f$  on their joint inputs while non-interactively communicating only with the server over a sequence of evaluations [11, 24, 39]. Such constructions have only been demonstrated in a setting where the clients and the server do not collude [39]. Our protocol achieves full security but requires an additional round of communication at the end due to decryption.

**Verifiable Polynomial Evaluations on Encrypted Data.** The second application in this area is verifiable computation on encrypted data. The notion was proposed by Gennaro et al. in [34] and follow-up works [12, 28, 29, 36] proposed constructions for computations such as linear functions and polynomial evaluations. These schemes provide both privacy of the outsourced data to the untrusted server and the integrity of the results computed by the server. However, these constructions rely on fully or somewhat homomorphic encryptions based on lattice and zero-knowledge proofs over polynomial rings, thus their overhead is high and they have not been realized in practice. Also these protocols cannot be directly extended to multi-clients.

Our scheme yields a more efficient verifiable computation on encrypted data for polynomial evaluations. The prover’s computation only involves operations on bilinear maps, making it one step closer to being practical. In the amortized setting, the verifier’s time is faster than evaluating the polynomial locally for multiple evaluations. In particular, to compute  $m$  evaluations on a degree- $d$  polynomial, the proof size is  $O(m + \log d)$  and the verifier’s time is  $O(d + m)$ .

Our model requires a setup phase for the clients prior to communicating with the server. This setup phase is independent of the input and is only carried out once, regardless of the number of polynomial evaluations computed later. The clients store a short state upon concluding this phase, which is later used to extract  $Q(x)$ . In our protocol, the parties run the key generation protocol for the underlying threshold encryption scheme, store the secret key share, and use it to partially decrypt the ciphertext returned from the server.

### 4.3 Non-interactive Two-party PSI (NISI)

Ishai et al. [44] introduced the Non-interactive Secure computation (NISC) model where, a Receiver first posts an “encryption” of its input publicly and then a Sender can compute a function over the encrypted input along with its input and obtain an “encryption” of the output that the Receiver can decrypt. The classic Yao’s garbled circuit based two-party protocol in the semi-honest setting when combined with a 2-round OT is an example of such a protocol. Several works have explored the feasibility and concrete efficiency of such protocols in the malicious Boolean setting [3, 5, 41, 44, 48]. Private polynomial commitments can be used directly to implement a non-interactive secure private set-intersection protocol by relying on a variant of the [31] protocol. Such a scheme will additionally have

the feature of reusability where the receiver only needs to post its encrypted input once and any number of senders can transmit the result of the set intersection to the receiver. An important application of reusable NISI is applicable is contact discovery in messaging services such as Signal and Telegram.

Concretely to PSI in the malicious setting, Cristofaro et al. [25] design a two-round PSI protocol with linear communication complexity. More recently, the work by Rosulek and Trieu [57] showed how to obtain a 2-round PSI by relying on a variant of the Diffie-Hellman Key Agreement and an ideal permutation oracle. This work has highly competitive communication and computation costs for small set sizes (between  $2^7$  and  $2^{16}$  elements). We provide a comparison of the communication costs in Table 5. We can see that our work is competitive in communication because the proofs are succinct in the batch setting. Additionally we rely on more standard assumptions. Even though our computation costs are higher our protocol could be useful in a client-server setting where the receiver is a lightweight client device and the sender is the server with significantly bigger computational resources. We further point out that the reported computational costs could be improved by further parallelizing our implementation. We leave this as future work to explore.

## 5 Implementation

We implemented our encrypted polynomial commitment scheme and the multi-party PSI scheme, and we present the experimental results in this section.

**Software and Hardware.** The system is implemented in C++. We use the ate-pairing library [1] for bilinear maps and the GMP library [2] for field arithmetic. Our experiments are executed on a BN-curve over a 254-bit prime, which offers 128-bit of security. There are 3200 lines of code for the encrypted polynomial commitment and 1000 lines for the other building blocks in the MPSI protocol. We ran all experiments on an AWS c5.9xlarge instance with an Intel Xeon Platinum 8000 processor and 72 GB of RAM. We report the average running time over 5 executions, except for the largest instances due to the long running time.

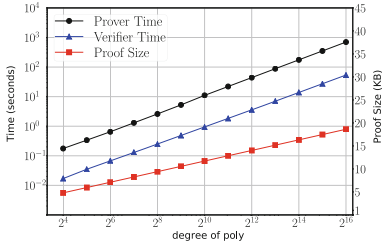
### 5.1 Private Polynomial Commitments

**Single Evaluation.** We first present the performance of our encrypted polynomial commitment scheme as a stand-alone primitive. Figure 8 shows the prover and verifier times (left  $y$ -axis) and proof size (right  $y$ -axis) of one evaluation

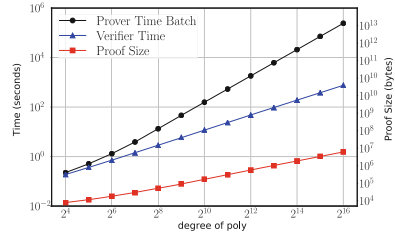
**Table 5.** Communication cost of two-party PSI with set size  $m$ .

$n$	$2^8$	$2^{16}$	$2^{20}$
[25]	62.74 (KB)	13.33 (MB)	213 (MB)
[57]	16.38 (KB)	4.19 (MB)	67.11 (MB)
Here (est.)	49.7 (KB)	5.86 (MB)	68 (MB)

of the variant with committed points (Sect. 2.2). We vary the degree of the polynomial from  $2^4$  to  $2^{16}$ . As shown in the figure, the prover time grows linearly with the polynomial degree. It takes 11 s to generate the proof for  $d = 2^{10}$  and 701 s to generate the proof for  $d = 2^{16}$ . The verifier time also grows linearly with the degree, as it has to update the commitment key together with the prover in our scheme. It takes 0.93 s to verify the proof for  $d = 2^{10}$  and 53.7 s for  $d = 2^{16}$ , which roughly matches the time on reducing the commitment key in the prover’s time. The proof size is only logarithmic on the degree of the polynomial and is very small in practice. It is 11.9 KB for  $d = 2^{10}$  and 18.6 KB for  $d = 2^{16}$ .



**Fig. 8.** Performance of single evaluation of our encrypted polynomial commitment with point hiding.



**Fig. 9.** Performance of multiple evaluations of our encrypted polynomial commitment with point hiding.  $m = d$ .

**Multiple Evaluations.** The major advantage of our scheme is the batched proofs for multiple evaluations and we further present the performance of evaluating multiple points in Fig. 9. In the figure, we set the number of evaluations the same as the degree of the polynomial, but our implementation supports both a larger degree and a larger number of evaluations. As shown in the figure, the prover time grows quadratically. It takes 0.225 s to generate a proof for  $m = d = 2^4$  and 242,395 s for  $m = d = 2^{16}$ .

The proof size and the verifier time are particularly good for multiple evaluations. The proof size is only 7.9 KB for  $m = d = 2^4$  evaluations and 6.1 MB for  $m = d = 2^{16}$  evaluations, which is significantly smaller than repeating the single evaluation protocol the same number of times. The experimental result matches the logarithmic complexity in  $d$  and the linear complexity in  $m$ .

The verifier time only grows quasi-linearly now. It only takes 757 s to verify  $2^{16}$  proofs of evaluations of a degree- $2^{16}$  polynomial, which is merely  $14\times$  larger than verifying a single proof. The experimental result justifies that the verifier time is amortized to  $O(\log d)$  for multiple evaluations and is particularly efficient in our application of multiparty PSI.

## 5.2 Performance of Multi-party PSI

In this section, we report the performance of our multiparty PSI protocol with malicious security. We executed all parties on the single AWS instance and we

**Table 6.** Total running time of our multiparty PSI scheme in seconds.

# of elements $m$	$2^8$	$2^{10}$	$2^{12}$	$2^{14}$	$2^{16}$
Size of bin $M$	$2^8$	$2^6$	$2^6$	$2^6$	$2^6$
# of bins $\mathcal{B}$	1	81	334	1,366	5,487
$n = 2$	13.94	130.01	536.1	2,192	8,264
$n = 8$	13.96	130.1	536.66	2,194	8,270
$n = 32$	13.97	130.4	538.4	2,199	8,292
$n = 128$	14.02	131.7	545.56	2,220	8,376
$n = 500$	14.26	136.4	562.76	2,301	8,712
$n = 1000$	14.58	142.9	589.5	2,410	9,141

simulated a network connection using the Linux `tc` command, communicating via a localhost network. We simulated a LAN setting with 10 Gbps network bandwidth. We executed  $P_2$  to  $P_n$  on the same machine but only count the running time of one of them in the total time. This is to better simulate the scheme in practice where all the parties can run the computation simultaneously.

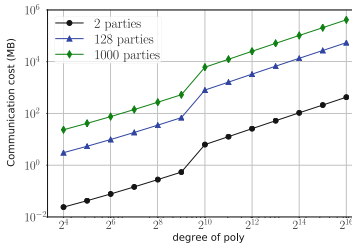
We tested our MPSI protocol for 2–1000 parties and  $2^8$ – $2^{16}$  elements per party (here we set  $m_{\max} = m_{\min}$ ) and the total running time are shown in Table 6. We applied the hashing technique described in Sect. 3 and the parameters achieving 40-bit of statistical security are included in the table.

As shown in the table, our protocol is slow for a small number of parties where it takes 13.94 s to compute a two-party intersection with  $2^8$  elements per party. This is  $55\times$  slower than the malicious MPSI scheme based on symmetric key primitives from [9, Table 5]. The gap is even larger on larger sets, which is expected as our protocol relies on public-key primitives. However, our running time hardly grew with the number of parties where it still takes 14.02 s for 128 parties with  $2^8$  elements each, and 14.58 s for 1000 parties. This is because most of the running time is due to evaluating the aggregated polynomial and generating the proofs using our commitment scheme, which only depends on the maximum size of the set  $m_{\max}$  and the size of  $P_1$ 's set  $m_{\min}$ . In contrast, the running time of PSimple [9] grows linearly with the number of parties and is 0.8 s for 32 parties with  $2^8$  elements each, which is  $17\times$  faster than ours. We expect that our protocol is faster than PSimple for 500 parties with  $2^8$  elements per party.

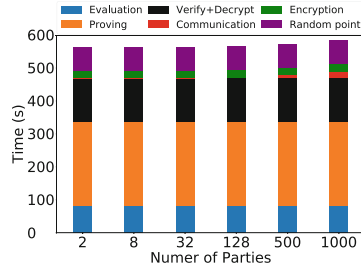
Our protocol is also efficient in communication. The total communication is shown in Fig. 10. As shown in the figure, the communication size for 2 parties with  $2^8$  elements per party is 279 KB, whereas the total communication for 1000 parties with  $2^8$  elements per party is 278 MB, which is not the bottleneck of our protocol. Compared with [9], the communication size is 7.5 MB for 2 parties and 7.5 GB for 1000 parties respectively, which is around  $27\times$  larger than ours. The jump in Fig. 10 for  $m = 2^{10}$  is due to using the hashing technique for  $m \geq 2^{10}$ .

We further show the breakdown of our total running time in Fig. 11. We fix the size of the set per party at  $2^{12}$  and vary the number of parties from 2 to 1000.

As shown in the figure, our protocol is clearly computation-heavy and most of the time is on the evaluations of the aggregated polynomial, the proof generation and the verification of our private polynomial commitment. Even with 1000 parties, they contribute to 97.5% of the total running time. Due of this observation, we could improve the total running time significantly through parallelization. Both the polynomial evaluations and the private polynomial commitment are trivially parallelizable. Moreover, the total running time of our scheme is not sensitive to the bandwidth of the network. On a WAN network with 100Mbps bandwidth, our scheme would become around two times slower for 1000 parties. By contrast, the performance of symmetric-key-based schemes such as PSimple is limited by the communication overhead. It cannot be improved through parallelization and will become worse on a network with lower bandwidth.



**Fig. 10.** Communication of our multi-party PSI protocol.



**Fig. 11.** Breakdown of the running time in our multiparty PSI protocol.  $m = 2^{12}$  elements per party.

Finally, another major advantage of our protocol is memory usage and scalability. As the memory usage of  $P_1$  is only  $O(m_{\max})$ , we are able to scale up to 1000 parties and  $2^{16}$  elements per party. The memory usage of  $P_1$  on this largest instance is only 1GB. We did not test more elements per party due to the long running time, but not have high memory usage. To compare, the PSimple scheme [9] runs out of memory for 12 parties and  $2^{20}$  elements per party. This is because  $P_1$  has to store random OTs for the garbled bloom filter with each party, which leads to a high overhead on the memory.

Overall, the experimental results show that our scheme has good scalability and communication in practice, and is particularly efficient for applications with a large number of parties or limited bandwidth networks.

**Acknowledgements.** We thank the anonymous PKC’23 reviewers for their helpful comments. The first and second authors are supported by ISF grant No. 1316/18. The second, third and fifth authors are supported by DARPA under Contract No. HR001120C0087. The third author was supported by Technology and Humanity Fund from Georgetown University’s McCourt School of Public Policy. 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.



## References

1. Ate pairing. <https://github.com/herumi/ate-pairing>
2. The GNU multiple precision arithmetic library. <https://gmplib.org/>
3. Abascal, J., Sereshgi, M.H.F., Hazay, C., Ishai, Y., Venkatasubramanian, M.: Is the classical GMW paradigm practical? the case of non-interactive actively secure 2pc. In: CCS, pp. 1591–1605 (2020)
4. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. *J. Cryptol.* **29**, 363–421 (2016)
5. Afshar, A., Mohassel, P., Pinkas, B., Riva, B.: Non-interactive secure computation based on cut-and-choose. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 387–404. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-55220-5\\_22](https://doi.org/10.1007/978-3-642-55220-5_22)
6. Backes, M., Datta, A., Kate, A.: Asynchronous computational VSS with reduced communication complexity. In: CT-RSA, vol. 7779, pp. 259–276 (2013)
7. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: CCS, pp. 863–874 (2013)
8. Bayer, S., Groth, J.: Zero-knowledge argument for polynomial evaluation with application to blacklists. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 646–663. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38348-9\\_38](https://doi.org/10.1007/978-3-642-38348-9_38)
9. Ben-Efraim, A., Nissenbaum, O., Omri, E., Paskin-Cherniavsky, A.: Psimple: practical multiparty maliciously-secure private set intersection. In: ASIA CCS, pp. 1098–1112 (2022)
10. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22792-9\\_7](https://doi.org/10.1007/978-3-642-22792-9_7)
11. Bhadauria, R., Hazay, C.: Multi-clients verifiable computation via conditional disclosure of secrets. In: SCN, pp. 150–171 (2020)
12. Bois, A., Cascudo, I., Fiore, D., Kim, D.: Flexible and efficient verifiable computation on encrypted data. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 528–558. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-75248-4\\_19](https://doi.org/10.1007/978-3-030-75248-4_19)
13. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_3](https://doi.org/10.1007/978-3-540-28628-8_3)
14. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: IEEE S&P, pp. 315–334 (2018)
15. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_24](https://doi.org/10.1007/978-3-030-45721-1_24)
16. Bünz, B., Maller, M., Mishra, P., Tyagi, N., Vesely, P.: Proofs for inner pairing products and applications. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13092, pp. 65–97. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92078-4\\_3](https://doi.org/10.1007/978-3-030-92078-4_3)
17. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: definitions and practical constructions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 262–288. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48800-3\\_11](https://doi.org/10.1007/978-3-662-48800-3_11)

18. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36362-7\\_5](https://doi.org/10.1007/978-3-642-36362-7_5)
19. Catalano, D., Fiore, D., Gennaro, R., Vamvourellis, K.: Algebraic (Trapdoor) one-way functions and their applications. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 680–699. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36594-2\\_38](https://doi.org/10.1007/978-3-642-36594-2_38)
20. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 371–389. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_21](https://doi.org/10.1007/978-3-662-44371-2_21)
21. Chase, M., Miao, P.: Private set intersection in the internet setting from lightweight oblivious PRF. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12172, pp. 34–63. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-56877-1\\_2](https://doi.org/10.1007/978-3-030-56877-1_2)
22. Chepurnoy, A., Papamanthou, C., Zhang, Y.: Edrax: a cryptocurrency with stateless transaction validation. IACR Cryptol. ePrint Arch., p. 968 (2018)
23. 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. LNCS, vol. 12105, pp. 738–768. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_26](https://doi.org/10.1007/978-3-030-45721-1_26)
24. Choi, S.G., Katz, J., Kumaresan, R., Cid, C.: Multi-client non-interactive verifiable computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 499–518. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36594-2\\_28](https://doi.org/10.1007/978-3-642-36594-2_28)
25. De Cristofaro, E., Kim, J., Tsudik, G.: Linear-complexity private set intersection protocols secure in malicious model. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 213–231. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17373-8\\_13](https://doi.org/10.1007/978-3-642-17373-8_13)
26. Fenske, E., Mani, A., Johnson, A., Sherr, M.: Distributed measurement with private set-union cardinality. In: CCS, pp. 2295–2312 (2017)
27. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: CCS, pp. 501–512 (2012)
28. Fiore, D., Gennaro, R., Pastro, V.: Efficiently encrypted data. In: ACM SIGSAC, pp. 844–855 (2014)
29. Fiore, D., Nitulescu, A., Pointcheval, D.: Boosting verifiable computation on encrypted data. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 124–154. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45388-6\\_5](https://doi.org/10.1007/978-3-030-45388-6_5)
30. 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](https://doi.org/10.1007/978-3-540-30576-7_17)
31. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24676-3\\_1](https://doi.org/10.1007/978-3-540-24676-3_1)
32. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: permutations over lagrange-bases for ocumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. **2019**, 953 (2019)
33. Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious key-value stores and amplification for private set intersection. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12826, pp. 395–425. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84245-1\\_14](https://doi.org/10.1007/978-3-030-84245-1_14)

34. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14623-7\\_25](https://doi.org/10.1007/978-3-642-14623-7_25)
35. Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 629–659. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70694-8\\_22](https://doi.org/10.1007/978-3-319-70694-8_22)
36. 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](https://doi.org/10.1007/978-3-642-40084-1_30)
37. Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: aggregating proofs for multiple vector communications. In: ACM SIGSAC, pp. 2007–2023 (2020)
38. Gordon, S.D., Hazay, C., Le, P.H.: Fully secure PSI via mpc-in-the-head. PoPETS **2022**(3), 291–313 (2022)
39. Gordon, S.D., Katz, J., Liu, F.-H., Shi, E., Zhou, H.-S.: Multi-Client verifiable computation with stronger security guarantees. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 144–168. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46497-7\\_6](https://doi.org/10.1007/978-3-662-46497-7_6)
40. Hazay, C.: Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 90–120. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46497-7\\_4](https://doi.org/10.1007/978-3-662-46497-7_4)
41. Hazay, C., Ishai, Y., Venkatasubramanian, M.: Actively secure garbled circuits with constant communication overhead in the plain model. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 3–39. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70503-3\\_1](https://doi.org/10.1007/978-3-319-70503-3_1)
42. Hazay, C., Lindell, Y.: Efficient oblivious polynomial evaluation with simulation-based security. IACR Cryptol. ePrint Arch., p. 459 (2009)
43. Hazay, C., Venkatasubramanian, M.: Scalable multi-party private set-intersection. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 175–203. Springer, Heidelberg (2017). [https://doi.org/10.1007/978-3-662-54365-8\\_8](https://doi.org/10.1007/978-3-662-54365-8_8)
44. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20465-4\\_23](https://doi.org/10.1007/978-3-642-20465-4_23)
45. Juels, A., Jr., B.S.K.: Pors: proofs of retrievability for large files. In: CCS, pp. 584–597 (2007)
46. 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](https://doi.org/10.1007/978-3-642-17373-8_11)
47. Lee, J.: Dory: efficient, transparent arguments for generalised inner products and polynomial commitments. IACR Cryptol. ePrint Arch. **2020**, 1274 (2020)
48. Mohassel, P., Rosulek, M.: Non-interactive secure 2PC in the offline/online and batch settings. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 425–455. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_15](https://doi.org/10.1007/978-3-319-56617-7_15)
49. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. **35**, 1254–1281 (2006)

50. Nguyen, D.T., Trieu, N.: Mpccache: privacy-preserving multi-party cooperative cache sharing at the edge. *IACR Cryptol. ePrint Arch.* (2021). <https://eprint.iacr.org/2021/317>
51. 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](https://doi.org/10.1007/978-3-642-36594-2_13)
52. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). [https://doi.org/10.1007/3-540-46766-1\\_9](https://doi.org/10.1007/3-540-46766-1_9)
53. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: SpOT-light: lightweight private set intersection from sparse OT extension. In: Boldyreva, A., Micciancio, D. (eds.) *CRYPTO 2019*. LNCS, vol. 11694, pp. 401–431. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_13](https://doi.org/10.1007/978-3-030-26954-8_13)
54. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: PSI from PaXoS: fast, malicious private set intersection. In: Canteaut, A., Ishai, Y. (eds.) *EUROCRYPT 2020*. LNCS, vol. 12106, pp. 739–767. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45724-2\\_25](https://doi.org/10.1007/978-3-030-45724-2_25)
55. Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based PSI with linear communication. In: Ishai, Y., Rijmen, V. (eds.) *EUROCRYPT 2019*. LNCS, vol. 11478, pp. 122–153. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17659-4\\_5](https://doi.org/10.1007/978-3-030-17659-4_5)
56. Raab, M., Steger, A.: “balls into bins” - a simple and tight analysis. In: *Randomization and Approximation Techniques in Computer Science*, pp. 159–170 (1998)
57. Rosulek, M., Trieu, N.: Compact and malicious private set intersection for small sets. *IACR Cryptol. ePrint Arch.*, p. 1159 (2021)
58. Schnorr, C.P.: Efficient signature generation by smart cards. *J. Cryptol.* **4**(3), 161–174 (1991). <https://doi.org/10.1007/BF00196725>
59. Setty, S.: Spartan: efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) *CRYPTO 2020*. LNCS, vol. 12172, pp. 704–737. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-56877-1\\_25](https://doi.org/10.1007/978-3-030-56877-1_25)
60. Tomescu, A., et al.: Towards scalable threshold cryptosystems. In: *IEEE S&P*, pp. 877–893 (2020)
61. Vlasov, A., Panarin, K.: Transparent polynomial commitment scheme with poly-logarithmic communication complexity. *IACR Cryptol. ePrint Arch.* **2019**, 1020 (2019)
62. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: *IEEE S&P*, pp. 926–943 (2018)
63. Wails, R., Johnson, A., Starin, D., Yerukhimovich, A., Gordon, S.D.: Stormy: statistics in tor by measuring securely. In: *CCS*, pp. 615–632 (2019)
64. Wieder, U.: Balanced allocations with heterogenous bins. In: *SPAA*, pp. 188–193 (2007)
65. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) *CRYPTO 2019*. LNCS, vol. 11694, pp. 733–764. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_24](https://doi.org/10.1007/978-3-030-26954-8_24)
66. Yuan, J., Yu, S.: Proofs of retrievability with public verifiability and constant communication cost in cloud. In: *SCC@ASIACCS*, pp. 19–26. ACM (2013)
67. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: *IEEE S&P* (2020)

68. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: VSQL: verifying arbitrary SQL queries over dynamic outsourced databases. In: IEEE S&P, pp. 863–880 (2017)
69. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: A zero-knowledge version of VSQL. IACR Cryptol. ePrint Arch. **2017**, 1146 (2017)



# Credibility in Private Set Membership

Sanjam Garg<sup>1,2</sup>, Mohammad Hajiabadi<sup>3</sup>, Abhishek Jain<sup>4</sup>, Zhengzhong Jin<sup>5</sup>,  
Omkant Pandey<sup>6</sup>, and Sina Shiehian<sup>7</sup>(✉)

<sup>1</sup> University of California, Berkeley, USA

<sup>2</sup> NTT Research, Sunnyvale, USA

<sup>3</sup> University of Waterloo, Waterloo, Canada

<sup>4</sup> Johns Hopkins University, Baltimore, USA

<sup>5</sup> MIT, Cambridge, USA

<sup>6</sup> Stony Brook University, Stony Brook, USA

<sup>7</sup> Snap Inc., Santa Monica, USA

shiaian@umich.edu

**Abstract.** A private set membership (PSM) protocol allows a “receiver” to learn whether its input  $x$  is contained in a large database  $\text{DB}$  held by a “sender”. In this work, we define and construct *credible private set membership* (*C-PSM*) protocols: in addition to the conventional notions of privacy, C-PSM provides a soundness guarantee that it is hard for a sender (that does not know  $x$ ) to convince the receiver that  $x \in \text{DB}$ . Furthermore, the communication complexity must be logarithmic in the size of  $\text{DB}$ .

We provide 2-round (i.e., round-optimal) C-PSM constructions based on standard assumptions:

- We present a black-box construction in the plain model based on DDH or LWE.
- Next, we consider protocols that support predicates  $f$  beyond string equality, i.e., the receiver can learn if there exists  $w \in \text{DB}$  such that  $f(x, w) = 1$ . We present two results with transparent setups: (1) A black-box protocol, based on DDH or LWE, for the class of  $\text{NC}^1$  functions  $f$  which are efficiently searchable. (2) An LWE-based construction for all bounded-depth circuits. The only non-black-box use of cryptography in this construction is through the bootstrapping procedure in fully homomorphic encryption.

As an application, our protocols can be used to build enhanced round-optimal leaked password notification services, where unlike existing solutions, a dubious sender *cannot* fool a receiver into changing its password.

## 1 Introduction

A two-party private set membership (PSM) protocol is an interactive protocol between a receiver holding an input  $x$  and a sender holding a database  $\text{DB}$ . The goal is that at the end of the interaction, the receiver only learns whether  $x \in \text{DB}$  while the sender learns nothing about  $x$ . Similar to private information retrieval [8], a desirable feature for PSM is efficiency of the receiver, which states

that the communication complexity and also the computational complexity of the receiver is sublinear (or more preferably logarithmic) in the size of DB. PSM and its closely related variant private set intersection (PSI) have found numerous applications such as contact discovery [17] and exposed password notification [2, 11, 16].

In the exposed password notification use-case, a user and a service provider run a PSM protocol to determine whether the user’s password is exposed in any leaked database. An often neglected aspect in this setting is whether the protocol provides a *credible* guarantee to the user that its password was actually leaked. In fact, a dubious sender might potentially keep falsely suggesting to the user that its password was exposed, causing the user to go through the process of updating its password.

A potential approach to enforce credibility might be requiring the sender to send its whole database in an encrypted format. It is plausible that such an approach, specially when implemented through protocols based on oblivious pseudorandom functions (OPRF) [2, 11], can provide credibility. However, sending the whole database would obviously make the protocol’s communication and the receiver’s computational complexity linear in the size of the database, and thus violates efficiency. Another approach may be using generic cryptographic succinct zero-knowledge arguments of knowledge. Such solutions incur an unsatisfactory computational overhead due to the use of *non-black-box* techniques. Therefore, we ask

*Can we construct asymptotically efficient black-box credible PSM protocols?*

## 1.1 Our Contributions

**Defining C-PSM.** In this work we initiate the study of *credibility* in PSM protocols. We define the notion of *credible private set membership* (C-PSM). Informally, a C-PSM for a relation  $\mathcal{R}$  is a two party protocol between a receiver and a sender where both the receiver and the sender have access to a common reference string (CRS). The receiver has an input  $x$  and the sender has a large database DB. The sender wants to convince the receiver that the database contains a witness  $w$  such that  $(x, w) \in \mathcal{R}$ . We require the following properties:

- The protocol consists of only two rounds.
- The communication and also the receiver’s computational complexity is at most logarithmic in the size of DB.
- The receiver’s input  $x$  remains hidden from the sender.
- The sender’s database remains private, i.e., a (malicious) receiver does not learn anything more than the fact that the database contains a valid witness.
- The protocol is sound, i.e., if the sender does not have a witness in the database, then, it is computationally hard for it to make the receiver accept.

We focus on black-box protocols, i.e., protocols which only make black-box use of their underlying cryptographic tools. For the soundness property to be meaningful and achievable in 2 rounds, we require the input  $x$  to have high entropy. Otherwise, if  $x$  is predictable, the sender can always include a valid witness for  $x$  in its database and convince the receiver. For the same reason we consider relations  $\mathcal{R}$  which are *instance entropic*. Roughly speaking, this means that any witness only satisfies a negligible fraction of instances. For example, the string equality relation is instance entropic.

**C-PSM for String Equality.** We start by considering the basic string equality relation, where the receiver wants to check if  $x \in \text{DB}$ . For this relation we construct a black-box 2-round C-PSM protocol in the plain model from either of the DDH or LWE assumption.

**Theorem 1 (Informal).** *Assuming the hardness of either of DDH or LWE, there exists a black-box 2-round C-PSM protocol in the plain model for the string equality relation.*

**C-PSM for Efficiently Searchable Relations.** We then turn to instance entropic relations beyond string equality. Specifically, we will consider the scenario where for some function  $f$ , the receiver wants to check whether DB contains  $c$  entries  $w_1, \dots, w_c$  such that  $f(x, \{w_i\}_{i \in [c]}) = 1$ . We first consider the class of *efficiently searchable* functions, i.e., functions which are in  $\text{NC}^1$ , and, for any input  $x$ , searching DB for witnesses can be implemented by a branching program of length logarithmic in DB. We construct a fully black-box 2-round C-PSM protocol for the class of *efficiently searchable* functions assuming either of DDH or LWE.

**Theorem 2 (Informal).** *Assuming the hardness of either of DDH or LWE, for every searchable function there exists a black-box 2-round C-PSM protocol with transparent setup.*

Next, we construct a C-PSM from LWE which is not restricted to efficiently searchable functions and supports all bounded-depth circuits. While this construction is not fully black-box, however, its non-black-use of cryptography is limited to the bootstrapping procedure in its underlying homomorphic encryption.

**Theorem 3 (Informal).** *Assuming the hardness of LWE, there exists a 2-round C-PSM protocol with transparent setup for every (bounded-depth) circuit. The only non-black-box use of cryptography in this C-PSM protocol is through bootstrapping in homomorphic encryption.*

We mention that all of our C-PSM protocols satisfy *statistical sender privacy*. This means that, our constructions guarantee the privacy of the sender even against computationally unbounded malicious receivers. Additionally, in our constructions which need a setup, receiver privacy is guaranteed even if the CRS is maliciously generated.



*Applications.* Our construction for string equality immediately gives a credible protocol for password exposure notification. In fact, since the C-PSM protocol in this construction only consists of two rounds, the receiver can publish its first message and wait for senders to inform him/her of a password exposure via C-PSM second message.

With our black-box construction for efficiently searchable relations, we can have protocols that perform more complicated tasks. For instance, consider a situation where the sender’s database consists of pairs of usernames and candidate passwords. A receiver wants to learn whether the database has an entry consisting of its username paired with a closely matching password (closely matching can for example mean having an edit distance no bigger than half the length of the password). We observe that our black-box construction supports this functionality. This is because given a username and password pair, the following branching program whose length is logarithmic in the size of the database can implement the corresponding search functionality:

1. First, search the database for an entry with a matching username. Note that this step can be implemented by a logarithmic length branching program through using the trie data structure.
2. Next, given an entry with a matching username, check whether the candidate password in the entry closely matches the input password. This step is independent of the size of the database and can be implemented by an  $NC^1$  circuit, and consequently by a polynomial sized branching program.

## 1.2 Related Work

The notion of zero-knowledge sets [19] allows a sender to convince a receiver whether an element exist in its database or not by sending a short proof. Our work differs from zero-knowledge sets in two aspects. First, we consider 2-round protocols whereas zero-knowledge sets consist of protocols having 3 rounds, where, in the first round the sender commits to its database and publishes a digest of this commitments. Second, there is no receiver privacy in zero-knowledge sets, i.e., the receiver sends its input in the clear.

A line of work [6, 7, 9] constructed concretely efficient *unbalanced* PSI protocols, i.e., PSI protocols where the sender’s set is considerably larger than the receiver’s set, from FHE. The PSI protocols constructed in these works provide sender privacy, receiver privacy and and communication sub-linear in the size of the sender’s set. While exposed password notification seems to be one of the main applications of the PSI protocols constructed in these works, however, they do not provide credibility. In fact [6] considers a heuristic approach to make it more difficult for a dubious sender to cheat. Roughly speaking, the proposal in [6] requires a sender to include the hash of the receiver’s input in the FHE ciphertext that it outputs. Then, it sets the FHE parameters such that it does not support computing this hash function. Our construction for string equality in Sect. 4 can be seen as a dual of this idea, where, we use the output of a one-way function as the input and treat the original input as the *label*. Unlike [6], we are able to formally prove the credibility of our construction.

Another work [15] considers oblivious polynomial evaluation (OPE). In this setting, the receiver wants to learn the image of its private input under a secret high-degree polynomial that is held by the sender. Notice that an instance of PSM can be converted into an instance of OPE where the degree of the polynomial is equal to the size of the database. The protocol in [15] provides receiver privacy, sender privacy and communication sub-linear in the degree of the polynomial. Additionally, this construction ensures that the evaluated value that receiver obtains truly corresponds to the polynomial that is held by the sender. While the latter property can be viewed as credibility, however, the way [15] enforces this property is by requiring the sender to send a commitment to its polynomial to the receiver. Consequently, this protocol needs three rounds.

### 1.3 Techniques

**C-PSM for String Equality.** We start by providing an overview of our C-PSM construction for the basic string equality functionality. Since we are aiming to keep the receiver’s complexity independent of the size of the database, it is natural to consider using homomorphic encryption (HE). However, a naive scheme where the receiver sends its input  $x$  encrypted under FHE, and the sender homomorphically searches its database, does not satisfy the properties of C-PSM:

- First and foremost, this construction is not credible because the sender can simply send a homomorphically encrypted positive answer regardless of its database.
- Furthermore, this construction does not provide sender privacy because homomorphic evaluation might reveal extra information about the sender’s database.

Our insight to solve the first issue is noticing that the receiver’s input has high entropy and therefore it is hard to invert its image under a one-way function. Specifically, the receiver, instead of sending an encryption of its input, sends an encryption of the image  $y = f(x)$  of its input under a one-way function  $f$ . The sender computes the images of all entries in its database under  $f$  and proceeds to homomorphically search these images for  $y$ . If found, the sender can homomorphically include the pre-image  $x$  in the ciphertext it sends to the receiver.

To add sender privacy, we will use a homomorphic encryption scheme with a property known in the literature as *malicious function privacy* [21]. Informally, this notion states that the evaluated ciphertexts reveal nothing beyond the value they are encrypting, and in particular they hide the function that was homomorphically evaluated. While the malicious function private HE construction in [21] makes extensive non-black-box use of cryptography, however, fortunately, we can instantiate the OT-based black-box HE construction in [14] with the recent rate-1 statistical sender private OT [1], which can be based on either LWE or DDH, to get a black-box malicious function private HE for branching programs.

**Beyond String Equality.** We now describe how we build a C-PSM supporting predicates beyond string equality. For the ease of exposition, we present a 4-round protocol and then briefly sketch how we compress it to 2 rounds. Recall that in this setting, the receiver holds an input  $x$  and the sender wants to convince the receiver that its database contains a witness  $w$  such that  $f(x, w) = 1$  for a specific predicate  $f$ . Our starting idea is to use homomorphic encryption for encrypting the receiver’s input, a black-box commit-and-prove system for committing to the sender’s database and generating zero-knowledge proofs, and Merkle trees [18] for creating a digest of this database. In more detail, similar to the string quality construction, the receiver encrypts its input under HE and sends the ciphertext to the sender. The sender then works as follows:

- First, it commits to the database using the commit and prove system, i.e., it secret shares each entry in the database and commits to these shares.
- Next, it hashes these commitments using a Merkle tree.
- Then, it homomorphically searches the database to find a valid witness  $w$  along with a Merkle hash opening for its corresponding commitment (or  $\perp$  if the database does not contain such a witness). Note that this does not involve any hash computations under the hood of HE. All hashes can be computed “outside,” and then moved to under the hood of HE.
- Next, the sender homomorphically generates the first prover message in the commit-and-prove system and sends it to the receiver.
- Finally, upon receiving a challenge from the receiver, the sender homomorphically opens a subset of the commitments produced in the first message and sends them to the receiver.

While this approach has succinct communication complexity, keeps the receiver’s input private, and is black-box thanks to the MPC-in-the-head [13] paradigm, however, it fails to protect against a malicious sender. In fact, a malicious sender whose database does not contain a valid witness can homomorphically cook up a database containing a witness and proceed to deceive the receiver. A straightforward approach to provide security against malicious senders is to require the sender to attach (in plain) a succinct non-interactive argument of knowledge (SNARK), showing that the evaluated ciphertext is the result of an honest evaluation using an actual database known by the sender. However, in addition to relying on unfalsifiable assumptions, this approach results in a very prohibitive solution and involves expensive non-black-box use of cryptography. For string equality we were able to overcome this issue by using deterministic encryption, but for richer functionalities this idea does not seem to be applicable. In summary, with the goal of avoiding expensive cryptography, the main challenge we face is “how do we tie the hands of a malicious sender to prevent it from cooking up a database under the hood of homomorphic encryption?”

**First Attempt: Attaching the Hash Root “Outside.”** Our first starting idea for tying the hands of the malicious sender is to have it send something “outside” the homomorphic encryption wrapper. The sender could cook up stuff

under homomorphic encryption but cannot do so outside! The receiver could then compare the information obtained under the hood of HE and check if it is consistent with the information provided “outside.” The hope is that given that a malicious sender cannot cook up stuff depending on receiver’s input “outside,” consistency is only possible if a valid witness exists in the database.

In particular, if we require the sender to include the root of the Merkle tree *in clear*, then, the homomorphic database cooking up attack that we described in the previous paragraph does not seem to work. Intuitively, the hash root seems to *bind* the prover to a database in clear, and if this database (and consequently the hash root) depends on the receiver’s input, then, a cheating prover has to somehow break the security of HE.

However, unfortunately, it is unclear how to prove security of this strategy. In other words, it is unclear how we could reduce the ability of the sender to break soundness to breaking the security of HE or the Merkle hash. A key issue is that the hash root does not have any *extractable information* to help with breaking the security of HE.

**Using SSB Hashing to Make a Random Point Extractable.** In order to fix the above issue, while avoiding expensive tools, we try for a very simple approach. In particular, we replace the generic Merkle Hash with a somewhere statistically binding (SSB) hash [12]. At a high level, SSB hash is a Merkle tree with an additional binding property. In more detail, in a SSB hash, the hashing key can be generated for binding to a specific position  $i$  in the input. The guarantee is that, the hash root now *statistically binds* to commitments to the value of the database at position  $i$ , which remains computationally hidden by the *index hiding* property. We assume a stronger *extractability* guarantee from our SSB hash. Namely, we assume that it is possible to *extract* the  $i$ th value given only the hash root and a *extraction trapdoor* which is generated along with the hashing key. Fortunately, these objects can be built based on any rate-1 OT using previous known techniques [12, 20].

Somewhat surprising, though with a subtle argument, this simple change allows us to reduce a malicious sender’s ability to cheat to break the security of HE or violate the index-hiding property of the SSB hash. We now sketch how using extractable SSB hashing we can reduce the security of HE to the soundness of C-PSM. Our reduction simply generates a SSB hash key binding to a *uniformly random* position and puts it in the CRS. First, observe that the index hiding property of SSB hash ensures that, during the execution, with noticeable probability, this random position is the same position that the cheating sender opens under the hood of HE. Clearly, if the adversary can somehow always avoid the random position encoded in the SSB hash key then that adversary can be used to break the index hiding property of SSB with probability better than a random guess. In the final step, we show a reduction that uses the value extracted from the SSB hash root — which from the prior step we know is correlated with the encrypted value under HE with a small probability — to directly break the security of HE.

**Instantiating HE.** Similar to our construction for string equality, we can use the malicious circuit private HE for branching programs that can be instantiated by combining [1] and [14]. For achieving compact communication complexity when using this instantiation of HE, searching the database for a witness should be implementable with a branching program whose length is logarithmic in the size of the database. That is, the predicate should be *efficiently searchable*. This is because in the [14] HE construction, the size of evaluated ciphertexts grow linearly in the length of the evaluated branching programs.

Another option is to use the LWE-based malicious circuit private HE in [10]. With this HE, our C-PSM construction can support every instance entropic predicate that can be implemented by a (bounded-depth) circuit. However, the HE in [10] is not fully black-box as it performs bootstrapping for every evaluation.

**Black-Box Commitment Generation.** A delicate issue is that, the sender algorithm, as currently described, would be non-black-box, because, generating the first prover message for the commit-and-prove system involves generating new commitments. We avoid this non-black-box step via the following trick: the sender generates many fresh commitments to 0 and 1 in the clear and then, obviously brings these fresh commitments under HE based on the message the prover commits to.

**4-Round to 2-Round.** Finally, we describe how to compress the described 4-round C-PSM to a 2-round protocol. To do this, the receiver sends its challenge via OT in the first round. In the second round, the sender prepares a C-PSM sender's message for each possible challenge and sends them to the receiver through OT response.

## 2 Preliminaries

We denote the security parameter by  $\lambda$ . For any  $\ell \in \mathbb{N}$ , we denote the set of the first  $\ell$  positive integers by  $[\ell]$ . For a set  $S$ ,  $x \leftarrow S$  denotes sampling a uniformly random element  $x$  from  $S$ . For a distribution  $D$ ,  $x \leftarrow D$  denotes sampling an element  $x$  from  $D$ .

### 2.1 Oblivious Transfer

We review the definition of rate-1 statistical sender private oblivious transfer.

**Definition 1 (Rate-1 Statistical Sender Private Oblivious Transfer).**

A (string) 1-out-of-2 OT consists of three algorithms:  $(\text{OT}_1, \text{OT}_2, \text{OT}_3)$ .

- $\text{OT}_1(1^\lambda, b)$ , on input a security parameter  $\lambda \in \mathbb{N}$  and a choice bit  $b \in \{0, 1\}$ , outputs a protocol message  $ot_1$  and a state  $st$ .
- $\text{OT}_2(ot_1, (m_0, m_1))$ , on input  $ot_1$ , and two sender inputs  $(m_0, m_1)$  of the same length, outputs a response  $ot_2$ .

–  $\text{OT}_3(st, ot_2)$ , on input a state  $st$  and  $ot_2$ , outputs a message  $m$ .

We require the following properties:

1. Correctness, for all security parameters  $\lambda$ , bits  $b \in \{0, 1\}$ , and sender inputs  $m_0, m_1 \in \{0, 1\}^*$ :

$$\Pr \left[ y = m_b \mid \begin{array}{l} (ot_1, st) \leftarrow \text{OT}_1(1^\lambda, b) \\ ot_2 \leftarrow \text{OT}_2(ot_1, (m_0, m_1)) \\ y \leftarrow \text{OT}_3(st, ot_2) \end{array} \right] = 1.$$

2. Receiver Security,  $ot \stackrel{c}{\approx} ot'$ , where  $(ot, *) \leftarrow \text{OT}_1(1^\lambda, 0)$  and  $(ot', *) \leftarrow \text{OT}_1(1^\lambda, 1)$ .
3. Statistical Sender Privacy, there exists an unbounded simulator  $\mathcal{S}$  such that for all (not necessarily honestly generated)  $ot_1$  there exists a bit  $b$ , such that for all sender inputs  $m_0, m_1 \in \{0, 1\}^*$ :

$$\text{OT}_2(ot_1, (m_0, m_1)) \stackrel{s}{\approx} \text{Sim}(1^\lambda, ot_1, m_b).$$

4. Rate-1: There exists a fixed polynomial  $\text{poly}$  such that for all polynomials  $n := n(\lambda)$ , for all first-round messages  $ot_1$  and for all  $(m_0, m_1) \in \{0, 1\}^n \times \{0, 1\}^n$ ,  $|ot_2| = n + \text{poly}(\lambda)$ , where  $ot_2 \leftarrow \text{OT}_2(ot_1, (m_0, m_1))$ .

**Theorem 4** ([1]). Assuming either DDH or LWE, there exists a black-box construction of rate-1 statistical sender private OT.

We also consider the following dual-mode variation of OT. Notice that this variation is not rate-1.

**Definition 2 (Dual-mode OT).** Let  $C$  be a constant. A 1-out-of- $C$  dual mode OT is a tuple of algorithms  $(\text{Setup}, \text{FakeSetup}, \text{Extract}, \text{OT}_1, \text{OT}_2, \text{OT}_3)$ , with the following syntax:

- $\text{Setup}(1^\lambda)$ , takes as input a security parameter, and outputs a  $crs$ .
- $\text{FakeSetup}(1^\lambda)$ , takes as input a security parameter, and outputs a  $crs_S$  and a trapdoor  $td$  that can be used to extract the sender's input.
- $\text{Extract}(td, ot_2)$ , takes as input the trapdoor  $td$ , and any  $\text{OT}_2$  message  $ot_2$ , outputs the sender's input  $\{m_c\}_{c \in C}$ .
- $\text{OT}_1, \text{OT}_2, \text{OT}_3$  have the same syntax as in Definition 1, except that they also take  $crs$  as input.

The correctness, receiver security and statistical sender privacy properties are the same as Definition 1. We additionally require the following properties:

1. *CRS Indistinguishability*, we have

$$crs \stackrel{c}{\approx} crs_S,$$

where  $crs$  is generated by  $\text{Setup}$ , and  $crs_S$  is generated by  $\text{FakeSetup}$ .

2. *Extraction Correctness*, for any receiver’s input  $b \in [C]$  and any unbounded adversary  $\mathcal{A}$ , we have

$$\Pr_{\substack{(crs_S, td) \leftarrow \text{FakeSetup}(1^\lambda), \\ (ot_1, st) \leftarrow \text{OT}_1(crs, b) \\ ot_2^* \leftarrow \mathcal{A}(crs, ot_1)}} [y \leftarrow \text{OT}_3(crs, st, ot_2^*), \{m_c^*\}_{c \in [C]} \leftarrow \text{Extract}(td, ot_2^*) : y = m_b^*] = 1.$$

**Theorem 5** ([22]). *Assuming hardness of either LWE or DDH, there exists a black-box construction of dual-mode oblivious transfer.*

### 2.2 Dual-Mode Commitments

We recall the definition of a dual-mode public key encryption system [22]. Since in our application the default mode these crypto systems are instantiated in is the *lossy* mode, we refer to them by *dual-mode commitments*.

**Definition 3.** *A dual-mode commitment is a tuple of PPT algorithms  $\text{Com} = (\text{Gen}, \text{FakeGen}, \text{Commit}, \text{Extract})$  having the following interface*

- $\text{Gen}(1^\lambda)$ , on input a security parameter  $\lambda$ , outputs a common reference string  $crs$ .
- $\text{FakeGen}(1^\lambda)$ , on input a security parameter  $\lambda$ , outputs a common reference string  $crs$  and an extraction trapdoor  $td$ .
- $\text{Commit}(crs, b)$ , on input a bit  $b \in \{0, 1\}$ , outputs a commitment  $com$ .
- $\text{Extract}(td, \tilde{t})$ , on input an extraction trapdoor  $td$ , and a commitment  $com$ , outputs a bit  $b \in \{0, 1\}$ .

We require the scheme to satisfy the following properties

1. *Extraction Correctness*, for any  $\lambda \in \mathbb{N}$  and  $b \in \{0, 1\}$ ,

$$\Pr[\text{Extract}(td, \tilde{t}) = b] = 1,$$

where,  $(crs, td) \leftarrow \text{Gen}(1^\lambda)$  and  $\tilde{t} \leftarrow \text{Commit}(crs, b)$ .

2. *Indistinguishable CRS Modes*, we have

$$\{crs : crs \leftarrow \text{Gen}(1^\lambda)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{crs : (crs, td) \leftarrow \text{FakeGen}(1^\lambda)\}_{\lambda \in \mathbb{N}}$$

3. *Statistical Hiding*, the following two distributions are statistically indistinguishable

$$\{\text{Commit}(crs, 0) : crs \leftarrow \text{Gen}(1^\lambda)\}_{\lambda \in \mathbb{N}} \stackrel{s}{\approx} \{\text{Commit}(crs, 1) : crs \leftarrow \text{Gen}(1^\lambda)\}_{\lambda \in \mathbb{N}}$$

**Theorem 6** ([22]). *Assuming hardness of either LWE or DDH, there exists a black-box construction of dual-mode commitments.*

### 2.3 Commit-and-Prove

We formulate the properties and the interface that we need from a commit-and-prove system. Then, we observe that the MPC-in-the-head paradigm can be used to build a commit-and-prove system with these properties.

**Definition 4.** *A commit-and-prove system with challenge space  $\mathcal{C}$  for a language  $L \in \text{NP}$ , is a tuple of algorithms  $\Pi = (\text{Setup}, \text{FakeSetup}, \text{Com}, \text{GenFresh}, \text{P1}, \text{P2}, \text{Verify}, \text{Extract})$  having the following interface*

- $\text{Setup}(1^\lambda)$ , on input a security parameter  $\lambda$ , outputs a common reference string  $crs$ .
- $\text{FakeSetup}(1^\lambda)$ , on input a security parameter  $\lambda$ , outputs a common reference string  $crs$  and an extraction trapdoor  $td$ .
- $\text{Com}(crs, w; r)$  on input a bitstring  $w \in \{0, 1\}^W$  outputs a commitment  $\tilde{w}$ .
- $\text{GenFresh}(crs)$ , on input a common reference string  $crs$ , outputs a sequence of fresh commitments along with their corresponding randomness  $\Gamma$ .
- $\text{P1}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma; r_P)$ , on input a common reference string  $crs$ , an instance  $x \in \{0, 1\}^\ell$ , a witness  $\mathbf{w} = \{w_i \in \{0, 1\}^W\}_{i \in [c]}$ , initial commitment randomness  $\mathbf{r} = \{r_i\}_{i \in [c]}$ , fresh commitments and their randomness  $\Gamma$ , and the random coins  $r_P$ , outputs the first part of proof string  $\pi_1$ .
- $\text{P2}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma, r_P, ch)$ , on input the same parameters of  $\text{P1}$ , the random coins used by  $\text{P1}$ , and the challenge  $ch$ , outputs the second part of the proof string  $\pi_2$ .
- $\text{Verify}(crs, x, \{\tilde{w}_i\}_{i \in [c]}, ch, \pi_1, \pi_2)$ , on input a common reference string  $crs$ , an instance  $x \in \{0, 1\}^\ell$ , a sequence of commitments  $\{\tilde{w}_i\}_{i \in [c]}$ , a challenge  $ch \in \mathcal{C}$ , and a proof string  $(\pi_1, \pi_2)$ , either accepts or rejects.
- $\text{Extract}(td, \tilde{t})$ , on input an extraction trapdoor  $td$ , and a commitment  $\tilde{t}$ , outputs a plaintext  $t \in \{0, 1\}^W$ .

We further require the commit and proof system to satisfy the following properties.

- *Completeness*, for any instance  $x \in L$ , and any tuple of strings  $(w_1, w_2, \dots, w_c) \in \{0, 1\}^{c \times W}$  which is a witness for  $x$ , let  $\tilde{w}_i \leftarrow \text{Com}(crs, w_i)$  be commitments to  $w_i$ , we have

$$\Pr_{\substack{crs \leftarrow \text{Setup}(1^\lambda) \\ \text{P1}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma) \\ ch \leftarrow \mathcal{C} \\ \pi_2 \leftarrow \text{P2}(ch, \text{st})}} [\text{Verify}(crs, x, \{\tilde{w}_i\}_{i \in [c]}, ch, \pi_1, \pi_2) \text{ accepts}] = 1.$$

- *Indistinguishable CRS modes*, we have

$$crs \stackrel{c}{\approx} crs',$$

where  $crs$  is generated by the genuine setup  $\text{Setup}(1^\lambda)$ , and  $crs'$  is generated by the fake setup  $\text{FakeSetup}(1^\lambda)$ .



- *Statistical Hiding*, for any two sequences of bitstrings  $w^0 = \{w^0\}_{\lambda \in \mathbb{N}}, w^1 = \{w^1\}_{\lambda \in \mathbb{N}}$ , the commitments are statistically indistinguishable under the *genuine setup*, namely,

$$\{\text{Com}(crs, w_\lambda^0) : crs \leftarrow \text{Setup}(1^\lambda)\}_{\lambda \in \mathbb{N}} \stackrel{s}{\approx} \{\text{Com}(crs, w_\lambda^1) : crs \leftarrow \text{Setup}(1^\lambda)\}_{\lambda \in \mathbb{N}}.$$

- $\epsilon$ -*Soundness*, let  $\mathcal{R}$  be the NP-relation for the language  $L$ . For any unbounded adversary  $(P1^*, P2^*)$ , after the following procedure,
  - Generate the fake CRS with trapdoor  $(crs, td) \leftarrow \text{FakeSetup}(1^\lambda)$
  - $(x, \{\tilde{w}_i\}_{i \in [c]}, \pi_1, \text{st}) \leftarrow P1^*(crs)$
  - Sample a random challenge  $ch \leftarrow \mathcal{C}$
  - $\pi_2 \leftarrow P2^*(ch, \text{st})$
 we have

$$\Pr[\mathcal{R}(x, \{\text{Extract}(td, \tilde{w}_i)\}_{i \in [c]}) \neq 1 \wedge \text{Verify}(crs, x, \{\tilde{w}_i\}_{i \in [c]}, ch, \pi_1, \pi_2) \text{ accepts}] < \epsilon.$$

- *Special Statistical Zero-Knowledge*, there exists a simulator algorithm  $\text{Sim}$ , such that, under any  $crs$  sampled by the genuine  $\text{Setup}$  algorithm, for any family of instances  $\{x_\lambda\}$  with  $x_\lambda \in L$ , any witness  $\{w_{\lambda,i}\}_{i \in [c]}$  for  $x_\lambda$ , any challenge  $ch \in \mathcal{C}$ , we have

$$(\text{Com}(crs, \{w_{\lambda,i}\}_{i \in [c]}; \mathbf{r}), \pi_1, \pi_2) \stackrel{s}{\approx} (c', \pi'_1, \pi'_2),$$

where  $\pi_1, \pi_2$  are the outputs of the honest prover’s algorithm for the instance  $x_\lambda$ , witness  $\{w_{\lambda,i}\}_{i \in [c]}$ , initial commitment randomness  $\mathbf{r}$ , and challenge  $ch$ , and  $(c', \pi'_1, \pi'_2) \leftarrow \text{Sim}(x_\lambda, ch)$  is output by the simulator.

**Theorem 7 (Black-Box Commit-and-Prove from MPC-in-the-Head).**

*There exists a commit-and-prove protocol with constant soundness error. Furthermore, the honest prover’s algorithms  $(P1, P2)$  only use information-theoretic building-blocks. Moreover, if the NP-relation of  $L$  can be verified by a circuit of depth  $d$ , then the algorithms  $P1, P2$  can also be computed by a circuit of depth  $O(d)$ .*

*Proof (Proof Sketch).* The work [13] constructed zero-knowledge from secure multiparty computation protocols. We use their zero-knowledge protocol to build a commit-and-prove system, and prove that it only makes black-box use of cryptography. We now describe the main algorithms.

- $\text{Com}(crs, w; r)$ : Let  $n = O(1)$  be a constant. First, it secret shares the witness  $w = w_1 \oplus w_2 \oplus \dots \oplus w_n$  to  $n$  shares, and then commits to each share separately using a dual-mode commitment scheme.
- $P1(crs, x, \mathbf{w}, \mathbf{r}, \Gamma; r_P)$ : Let  $R(\cdot, \cdot)$  be the relation circuit of the language  $L$ . It uses a semi-honest information theoretic multiparty computation scheme (MPC) in the dishonest majority setting [4] for  $n$  parties. For every  $i \in [n]$ , the  $i$ th party holds  $w_i$  as its input. The prover runs the MPC “in its head” to jointly compute  $R(x, w_1 \oplus w_2 \oplus \dots \oplus w_n) = 1$ , and obtains the view of each party  $\text{View}_1, \text{View}_2, \dots, \text{View}_n$ . Then, it outputs commitments to the views.

- $ch \leftarrow \mathcal{C}$ : The challenge  $ch$  represent two random parties  $ch \leftarrow [n] \times [n]$ .
- $\text{P2}(crs, x, \mathbf{w}, \mathbf{r}, \Gamma, r_P, ch)$ : The prover does the same computation as  $\text{P1}$ , and then opens the commitment of the views specified by  $ch$ , and also opens the commitments to the shares specified by  $ch$ .
- **Verify**: The verifier checks
  - The openings of the commitments are correct.
  - The views are consistent. Namely, the messages sent and received have the same values.

The zero-knowledge and the soundness property follow from the security and the correctness of the underlying MPC scheme. Now, we show that the construction only makes black-box use of cryptography. Since the MPC is information theoretic, the only part that uses cryptography is the commitments in  $\text{P1}$ . To make  $\text{P1}$  information theoretic, we provide it a series of fresh commitments to 0 and 1 and their randomness in  $\Gamma$ . Then we have the prover choose which commitment it needs to use. This makes  $\text{P1}$  information theoretic.

Now we analyze the depth of  $\text{P1}$ . Let the depth of the circuit  $R$  be  $d$ . Since we only have a constant number of parties, the secret sharing of  $\mathbf{w}$  needs a constant depth circuit. For each gate in  $R$ , we only need a constant depth circuit to compute the corresponding messages in the MPC. Hence, the computation of the views  $\text{View}_1, \text{View}_2, \dots, \text{View}_n$  can be done in depth  $O(d)$ .

The depth of  $\text{P2}$  can also be bounded by  $O(d)$ . This is because it does the same computation as  $\text{P1}$ , and an additional commitment opening in the end. The commitment opening is selecting the commitment randomness specified by  $ch$ . Hence, it can be computed by a constant depth circuit.

## 2.4 Maliciously Function Private Homomorphic Encryption

We review the definition of maliciously function private homomorphic encryption. Notice that in our abstraction of homomorphic encryption, secret keys are generated corresponding to fresh ciphertexts, and can only decrypt the evaluated versions of their corresponding fresh ciphertexts. The reason we choose this abstraction is that we want it to be consistent with the construction in [14]. We mention that this abstraction is sufficient for our use-case.

**Definition 5** ([21]). *Let  $\mathcal{F} = \{\mathcal{F}_{\lambda,L}\}_{\lambda,L \in \mathbb{N}}$  be a family of boolean functions, where for each  $\lambda, L \in \mathbb{N}$ , the functions in  $\mathcal{F}_{\lambda,L}$  have input size  $\ell(\lambda, L)$ . A maliciously function private homomorphic encryption (HE) scheme for  $\mathcal{F}$  is a tuple of algorithms*

$\text{HE} = (\text{Enc}, \text{Eval}, \text{Dec}, \text{Sim})$ , *where, except for Sim the rest of the algorithms are PPT, having the following interfaces*

- $\text{Enc}(1^\lambda, 1^L, m)$ , *given a security parameter  $\lambda \in \mathbb{N}$ , a function family index  $L \in \mathbb{N}$ , and a message  $m \in \{0, 1\}^\ell$ , outputs a ciphertext  $ct \in \{0, 1\}^{\ell_{ct}(\lambda, L)}$  and a private key  $sk$ .*
- $\text{Eval}(ct, f)$ , *given a ciphertext  $ct$ , and a boolean function  $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ , outputs an evaluated ciphertext  $ct_{eval} \in \{0, 1\}^{\ell_{eval}}$ .*

- $\text{Dec}(sk, ct)$ , given a secret key  $sk$  and a ciphertext  $ct$ , outputs a bit  $b \in \{0, 1\}$ .
- $\text{Sim}(ct^*, b)$ , on input a ciphertext  $ct^* \in \{0, 1\}^{\ell_{ct}(\lambda, L)}$ , and a bit  $b$ , outputs a simulated ciphertext  $ct_{sim}$ .

We consider HE schemes that satisfy the following properties:

1. Completeness, for every  $\lambda, L \in \mathbb{N}$ , every function  $f \in \mathcal{F}_{\lambda, L}$  and every input  $m \in \{0, 1\}^\ell$ ,

$$\Pr[\text{Dec}(sk, ct_{eval}) = f(m)] = 1,$$

where,  $(ct, sk) \leftarrow \text{Enc}(1^\lambda, 1^L, m)$ , and  $ct_{eval} \leftarrow \text{Eval}(ct, f)$ .

2. Compactness, there exists a fixed polynomial  $\ell_{eval} = \ell_{eval}(\lambda, L)$  such that evaluated ciphertexts have size  $\ell_{eval}(\lambda, L)$ , i.e., the size of evaluated ciphertexts only depend on the index of the family of functions being evaluated.
3. Semantic Security, for every non-uniform polynomial-size adversary  $\mathcal{A}$ , every  $L \in \mathbb{N}$ , and every two sequence of message  $m^0 = \{m_\lambda^0 \in \{0, 1\}^{\ell(\lambda, L)}\}_{\lambda \in \mathbb{N}}$  and  $m^1 = \{m_\lambda^1 \in \{0, 1\}^{\ell(\lambda, L)}\}_{\lambda \in \mathbb{N}}$  the probabilities

$$\Pr[\mathcal{A}(ct) = 1], \tag{1}$$

in the following two experiments differ by only  $\text{negl}(\lambda)$ :

- in experiment 0,  $(ct, sk) \leftarrow \text{Enc}(1^\lambda, 1^L, m_\lambda^0)$
- in experiment 1,  $(ct, sk) \leftarrow \text{Enc}(1^\lambda, 1^L, m_\lambda^1)$

4. Malicious Function Privacy, for every  $L \in \mathbb{N}$ , and every ciphertext  $ct^* \in \{0, 1\}^{\ell_{ct}(\lambda, L)}$ , there exists a  $m^* \in \{0, 1\}^{\ell(\lambda, L)}$  such that, for every function  $f \in \mathcal{F}_{\lambda, L}$ ,

$$\text{Eval}(ct^*, f) \stackrel{s}{\approx} \text{Sim}(ct^*, f(m^*))$$

.

If we instantiate the rate-1 OT-based HE construction of [14] with the recent rate-1 statistical sender private OT of [1] we get a malicious function private HE for branching programs.

**Theorem 8** ([1, 14]). *Assuming either DDH or LWE, there exists a black-box construction of maliciously function private homomorphic encryption scheme for the function family  $\mathcal{B} = \{\mathcal{B}_L\}_{L \in \mathbb{N}}$ , where for each  $L \in \mathbb{N}$ ,  $\mathcal{B}_L$  is the set of branching programs of length  $L$ .*

If we slightly relax the black-box requirement, we can have a lattice-based leveled maliciously function private FHE scheme, i.e., a maliciously function private HE scheme supporting all bounded-depth polynomial circuits.

**Theorem 9** ([10]). *Assuming LWE, there exists a leveled maliciously function private homomorphic encryption scheme. The non-black-box use of cryptography in this scheme is restricted to bootstrapping (which is needed for every evaluation).*

## 2.5 Somewhere Statistically Binding Hash

Here we define a variant of somewhere statistically binding hashes [12].

**Definition 6.** Fix a word size  $W = W(\lambda)$ . A somewhere statistical binding hash scheme is a tuple of PPT algorithms  $\text{SSB} = (\text{Gen}, \text{Hash}, \text{Verify}, \text{Extract})$  with the following syntax.

- $\text{Gen}(1^\lambda, N, S)$ , on input a security parameter  $\lambda$ , a database size  $N$ , and a subset of indices  $S \subseteq [N]$ , outputs a hash key  $hk$  along with a trapdoor  $td$ .
- $\text{Hash}(hk, \text{DB})$ , on input a hash key  $hk$  and a database  $\text{DB}$  of  $N$  words of size  $W$ , outputs a hash value  $h$  along with  $N$  openings  $\{\tau_i\}_{i \in [N]}$ .
- $\text{Verify}(hk, h, i, x, \tau)$ , on input a hash key  $hk$ , a hash value  $h$ , an index  $i$ , a word  $x$ , and an opening  $\rho$ , either accepts or rejects.
- $\text{Extract}(td, h)$ , on input a hash value  $h$ , and a trapdoor  $td$ , outputs entries  $\{x_i\}_{i \in S}$ .

We require the scheme to satisfy the following properties:

1. Correctness, for all  $\lambda, N \in \mathbb{N}$ , any subset of indices  $S \subseteq [N]$ , any index  $i \in [N]$ , and any database  $\text{DB}$  of size  $N$ , we have

$$\Pr[\text{Verify}(hk, h, i, \text{DB}_i, \tau_i) \text{ accepts}] = 1,$$

where,  $(hk, td) \leftarrow \text{Gen}(1^\lambda, N, S)$  and  $(h, \{\tau_i\}_{i \in [N]}) := \text{Hash}(hk, \text{DB})$ .

2. Index Hiding, for any two sets  $S_1, S_2$  of the same size, we have

$$\text{crs}_1 \stackrel{c}{\approx} \text{crs}_2,$$

where  $\text{crs}_1$  is generated by  $\text{Gen}(1^\lambda, N, S_1)$ , and  $\text{crs}_2$  is generated by  $\text{Gen}(1^\lambda, N, S_2)$ .

3. Extraction Correctness, for all  $\lambda, N \in \mathbb{N}$ , any subset of indices  $S \subseteq [N]$ , any index  $i \in [N]$ , any database  $\text{DB}$  of size  $N$ , and any hash  $h$ , we have

$$\Pr[\text{Verify}(hk, h, i, \text{DB}_i, \tau_i) \text{ accepts} \wedge x_i \neq \text{DB}_i] = 0,$$

where,  $(hk, td) \leftarrow \text{Gen}(1^\lambda, N, S)$  and  $\{x_i\}_{i \in [S]} := \text{Extract}(td, h)$ .

4. Efficiency: any hash key  $hk$  and opening  $\tau$  corresponding to size  $N$  databases and index sets of size  $|S|$ , are of size  $|S| \cdot \log(N) \cdot \text{poly}(\lambda)$ . Further,  $\text{Verify}$  can be implemented by a circuit of size  $|S| \cdot \log(N) \cdot \text{poly}(\lambda)$ .

Our definition is slightly stronger than the one in [12] in that (i) our hashing key is binding to a subset of indices instead of binding to a single index and, (ii) we need perfect extractable binding instead of just statistical binding, i.e., there is a trapdoor that allows extracting the  $i$ th value for each binding index  $i$ . We can get the former property by repeating any single-index binding scheme multiple times in parallel. For the latter property, we notice that the HE-based construction in [12] already achieves this property, however, it is non-black-box due to the use of bootstrapping in the underlying HE. We observe that if we use a rate-1 OT scheme instead of HE, then, we have a black-box construction satisfying all the requirements in Definition 6. Please refer to the full version for a sketch of the construction.

**Theorem 10.** *Assuming hardness of either DDH or LWE, there exists a black-box construction of somewhere statistically binding hash satisfying the properties listed in Definition 6.*

### 3 Defining C-PSM

First, we formally define the relations we consider in our protocols.

**Definition 7 (H-Instance Entropic Relations).** *Let  $X$  and  $Y$  be two sets. Let  $\mathcal{R} \subseteq X \times Y$  be a relation. For any distribution  $D$  on  $X$ , we say  $\mathcal{R}$  is  $H$ -instance entropic with respect to  $D$ , if, for every  $w \in Y$ ,*

$$\Pr_{x \leftarrow D} [(x, w) \in \mathcal{R}] \leq 2^{-H}.$$

Next, we define the search functionality.

**Definition 8 (Search function).** *Fix parameters  $\ell, c, W, N \in \mathbb{N}$ . The procedure `Search` takes as input a boolean function  $f : \{0, 1\}^\ell \times \{0, 1\}^{c \cdot W} \rightarrow \{0, 1\}$ , a bitstring  $x \in \{0, 1\}^\ell$ , and a database  $\text{DB}$  consisting of  $N$  words of size  $W$ . It either outputs the lexicographically first  $c$  indices  $i_1, \dots, i_c \in [N]$  such that  $f(x, \text{DB}_{i_1}, \dots, \text{DB}_{i_c}) = 1$  or  $\perp$  if no such  $c$  indices exist.*

We are now ready to define C-PSM.

**Definition 9 (2-Round C-PSM).** *Let  $\ell = \ell(\lambda), c = c(\lambda), W = W(\lambda)$  and  $H = H(\lambda)$  be integer parameters. Let  $D$  be a distribution on  $\{0, 1\}^\ell$ . Fix a family of  $H$ -instance entropic boolean functions  $f = \{f_\lambda : \{0, 1\}^{\ell(\lambda)} \times \{0, 1\}^{c(\lambda) \cdot W(\lambda)} \rightarrow \{0, 1\}\}$  with respect to  $D$ . A credible private set membership protocol for  $f$ , denoted by C-PSM, is a protocol between a sender and a receiver described by a tuple of PPT algorithms (`Setup`, `R`, `S`, `Verify`), with the following syntax:*

- `Setup`( $1^\lambda, N$ ), on input a security parameter  $\lambda$  and database size  $N$ , outputs a CRS  $\text{crs}$ .
- `R`( $\text{crs}, x$ ), given a CRS  $\text{crs}$  and an input  $x$ , outputs a receiver message  $\alpha$  and an internal state  $st$ .
- `S`( $\text{crs}, \alpha, \text{DB}$ ), on input a CRS  $\text{crs}$ , receiver message  $\alpha$ , and database  $\text{DB}$ , outputs a sender message  $\beta$ .
- `Verify`( $\beta, st$ ), on input a sender message  $\beta$  and internal state  $st$ , either accepts or rejects.

We require the protocol to satisfy the following properties

1. Correctness, for every  $\lambda, N \in \mathbb{N}$ , every input  $x \in \{0, 1\}^\ell$ , and every database  $\text{DB}$  of size  $N$  such that  $\text{Search}(f, x, \text{DB}) \neq \perp$ , we have

$$\Pr_{\substack{\text{crs} \leftarrow \text{Setup}(1^\lambda, N) \\ (\alpha, st) \leftarrow \text{R}(\text{crs}, x) \\ \beta \leftarrow \text{S}(\text{crs}, \alpha, \text{DB})}} [\text{Verify}(\beta, st) \text{ accepts}] = 1.$$

2.  $\delta$ -Soundness, for every non-uniform malicious sender  $S^* = \{S_\lambda^*\}_{\lambda \in \mathbb{N}}$ , and every  $\lambda, N \in \mathbb{N}$ ,

$$\Pr_{\substack{crs \leftarrow \text{Setup}(1^\lambda, N) \\ x \leftarrow D \\ (\alpha, st) \leftarrow R(crs, x) \\ \beta \leftarrow S^*(crs, \alpha)}}} [\text{Verify}(\beta, st) \text{ accepts}] \leq \delta(\lambda) + 2^{-H(\lambda)}$$

3. Receiver Privacy, for any sequence of CRS strings  $crs = \{crs_\lambda\}_{\lambda \in \mathbb{N}}$ , and for any two sequence of input strings  $x^0 = \{x_\lambda^0\}_{\lambda \in \mathbb{N}}$ ,  $x^1 = \{x_\lambda^1\}_{\lambda \in \mathbb{N}}$ ,

$$\{crs_\lambda, \alpha : (\alpha, st) \leftarrow R(crs_\lambda, x_\lambda^0)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{crs_\lambda, \alpha : (\alpha, st) \leftarrow R(crs_\lambda, x_\lambda^1)\}_{\lambda \in \mathbb{N}}.$$

4. Statistical Malicious Sender Privacy, there is a (possibly unbounded) simulator algorithm  $\text{Sim}$ , such that, for every sequence of first message strings  $\alpha = \{\alpha_\lambda\}_{\lambda \in \mathbb{N}}$ , there exists a sequence of inputs  $x^* = \{x_\lambda^*\}$ , such that for any  $N \in \mathbb{N}$ , and for every database  $\text{DB}$  of  $N$  records, the following two distributions are statistically indistinguishable,

- first, generate  $crs \leftarrow \text{Setup}(1^\lambda, N)$ , output  $\text{Sim}(crs_\lambda, \alpha_\lambda, x_\lambda^*, \text{Search}(f, x_\lambda^*, \text{DB}))$ ,
- first, generate  $crs \leftarrow \text{Setup}(1^\lambda, N)$ , output  $S(crs, \alpha_\lambda, \text{DB})$ .

5. Efficiency, both  $R$  and  $\text{Verify}$  have runtime  $\text{poly}(\lambda, \ell, c, W, \log(N))$ .

*Remark 1.* Notice that the notion of sender privacy in in Definition 9 does not prevent leaking the indices for the witness in the database. This is W.L.O.G and merely for the ease of exposition. To prevent this leakage, the sender can simply randomly shuffle the entries in its database.

## 4 Construction for String Equality

Here we present the simplest version of our construction where the predicate is simply string equality, that is, the receiver wants to learn whether its input is in the sender's database. The resulting protocol has 2 rounds, achieves  $\text{negl}(\lambda)$ -soundness in a single repetition, and, does not depend on a CRS. For this construction, let the input size and the database word size be equal, i.e.,  $\ell(\lambda) = W(\lambda) \geq \lambda$ . Also, define  $D$  to be the uniform distribution on  $\{0, 1\}^\ell$ . Observe that for strings of length  $\ell$ , the string equality relation is an  $\ell$ -instance entropic relation with respect to  $D$ .

We new describe the ingredients in our construction.

- The first ingredient is a one-way function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ . We assume  $f$  maps  $\ell(\lambda)$ -bit inputs to  $m(\lambda)$ -bit outputs.
- The second ingredient is a maliciously circuit private homomorphic encryption scheme  $\text{HE} = (\text{Enc}, \text{Eval}, \text{Dec}, \text{Sim})$  for the class of branching programs  $\mathcal{B} = \{\mathcal{B}_L\}_{\lambda, L \in \mathbb{N}}$ . Where, for each  $L \in \mathbb{N}$ ,  $\mathcal{B}_L$  consists of all branching programs of length  $L$ .

**Construction 1.** Let  $L := L(\lambda, N)$  be the length of the branching program computing the function  $\text{Find}$  described in Fig. 1. The construction is as follows:

- $R(x)$ :
  - Compute the image of  $x$  under  $f$  to obtain  $y := f(x)$ .
  - Encrypt  $y$  under HE to produce  $(ct, sk) \leftarrow \text{HE.Enc}(1^\lambda, 1^L, y)$ .
  - Output  $\alpha := ct$  and store internal state  $st := sk$ .
- $S(\alpha, \text{DB})$ :
  - Parse  $\alpha := ct$ .
  - Apply  $f$  to every entry in DB to obtain  $\widetilde{\text{DB}} = \{\widetilde{\text{DB}}_i := f(\text{DB}_i)\}_{i \in [N]}$ .
  - Homomorphically evaluate the function  $\text{Find}_{\widetilde{\text{DB}}, \text{DB}}$  on  $ct$  to obtain  $ct_{eval} \leftarrow \text{HE.Eval}(ct, \text{Find}_{\widetilde{\text{DB}}, \text{DB}})$ .
  - Output  $\beta := ct_{eval}$ .
- $\text{Verify}(\beta, st)$ :
  - Parse  $\beta$  and  $st$  as  $\beta = ct_{eval}$  and  $st = sk$  respectively.
  - Decrypt  $ct_{eval}$  to obtain  $\tilde{x} := \text{HE.Dec}(sk, ct_{eval})$ .
  - Accept iff  $f(\tilde{x})$  equals  $f(x)$ .

```

procedure Find $_{\widetilde{\text{DB}}, \text{DB}}(y)$ 
  if  $y \notin \widetilde{\text{DB}}$  then
    Output  $\perp$ 
  else
    Find the smallest index  $i$  such that  $\widetilde{\text{DB}}_i = y$ .
    Output  $\text{DB}_i$ .

```

**Fig. 1.** Description of the labeled-PSM functionality Find

Correctness and receiver privacy of Construction 1 immediately follows from the correctness and semantic security of HE. For efficiency, we have to argue that the length of the branching program computing Find is logarithmic in  $N$ . To do this, as shown in [5], we can convert the database DB to a trie, and essentially implement Find by a branching program of length  $\ell$ .

We now prove the soundness of Construction 1.

**Theorem 11.** *Assuming  $f$  is one-way, Construction 1 is  $\text{negl}(\lambda)$ -sound.*

*Proof.* Let  $S^*$  be a malicious sender. Denote the success probability of  $S^*$  by  $p$ . In more detail,  $p$  is defined as

$$p := \Pr_{\substack{x \leftarrow D \\ (ct, sk) \leftarrow \text{HE.Enc}(1^\lambda, 1^L, f(x)) \\ \beta \leftarrow S^*(ct) \\ \tilde{x} := \text{HE.Dec}(sk, \beta)}} [f(\tilde{x}) = f(x)].$$

We use  $S^*$  to build a PPT adversary  $\mathcal{A}$  which breaks the one-wayness of  $f$  with probability  $p$ .  $\mathcal{A}$  works as follows, on input an image  $y$ , it first encrypts  $y$  by HE to obtain  $(ct, sk) \leftarrow \text{HE.Enc}(1^\lambda, 1^L, y)$ . It then runs  $S^*$  on input  $ct$  to get  $ct_{eval} \leftarrow S^*(ct)$ . Finally,  $\mathcal{A}$  decrypts  $ct_{eval}$  using  $sk$  and outputs  $\tilde{x} := \text{HE.Dec}(sk, ct_{eval})$  as

the preimage of  $y$ . Now observe that as long as  $y$  is an image of an input chosen from the distribution  $D$ , the view of  $S^*$  when interacting with  $\mathcal{A}$  is identical to its view in the soundness game. Therefore,

$$\Pr_{\substack{x \leftarrow D \\ y := f(x) \\ \tilde{x} \leftarrow \mathcal{A}(y)}} [f(\tilde{x}) = f(x)] = p.$$

This completes the proof.

**Theorem 12.** *Assuming HE is maliciously circuit private, Construction 1 satisfies statistical malicious sender privacy.*

*Proof.* Let  $\alpha$  be an arbitrary first message and DB be any database of size  $N \in \mathbb{N}$ . We only describe the simulator algorithm  $\text{Sim}$ , the theorem follows instantly from the malicious function privacy of HE.

- $\text{Sim}$  receives as input a first message  $\alpha := ct$ , and a bitstring  $x^*$ .
- Using the HE simulator it computes  $ct_{eval} \leftarrow \text{HE.Sim}(ct, x^*)$ .
- It outputs  $ct_{eval}$ .

## 5 Construction for Predicates Beyond String Equality

Now we consider richer families of predicates. Fix input length  $\ell = \ell(\lambda)$ , word size  $W = W(\lambda)$ , function arity  $c = c(\lambda)$ , distribution  $D$  on  $\{0, 1\}^\ell$ , and entropy parameter  $H = H(\lambda)$ . Let  $f : \{0, 1\}^\ell \times \{0, 1\}^{c \cdot W} \rightarrow \{0, 1\}$  be an  $H$ -instance entropic function with respect to  $D$ .

In the rest of the paper, we construct a 2-round C-PSM protocol in three steps.

- First, we construct a 4-round protocol satisfying a weaker notion of soundness, where, it is only required that an adversary cannot convince a verifier for any *fixed* set of indices.
- Then, using dual-mode 2-round OT, we show how to compress the 4-round protocol to a 2-round protocol which still has weak soundness.
- Finally, we amplify the soundness of the 2-round protocol by parallel repetition to achieve a (strongly) sound 2-round protocol.

### 5.1 Weakly-Sound 4-Round Protocol

We first construct a *weakly-sound* 4-round protocol with constant soundness. Where a weakly-sound 4-round C-PSM protocol is defined as follows:

**Definition 10 (Weakly-Sound 4-Round C-PSM).** *A credible private set membership protocol with challenge space  $\mathcal{C}$  for  $f$  is a protocol between a sender and a receiver described by a tuple of PPT algorithms  $(\text{Setup}, R, S1, S2, \text{Verify})$ , with the following syntax:*

- $\text{Setup}(1^\lambda, N)$ , on input a security parameter  $\lambda$  and database size  $N$ , outputs a CRS crs.



- $R(crs, x)$ , given a CRS  $crs$  and an input  $x$ , outputs a receiver message  $\alpha$  and an internal state  $st_R$ .
- $S1(crs, \alpha, DB)$ , on input a CRS  $crs$ , a receiver message  $\alpha$ , and a database  $DB$ , outputs a sender message  $\beta_1$  and an internal state  $st_S$ .
- $S2(crs, ch, st_S)$ , on input a CRS  $crs$ , a challenge  $ch$ , and an internal state  $st_S$ , outputs a sender message  $\beta_2$ .
- $Verify(\beta_1, ch, \beta_2, st_R)$  on input sender messages  $\beta_1, \beta_2$ , challenge  $ch$ , and internal state  $st_R$ , either accepts and outputs a sequence  $S = \{i_k\}_{k \in [c]}$  of indices, or, rejects.

We require the protocol to satisfy the following properties

1. Correctness, for every  $\lambda, N \in \mathbb{N}$ , every input  $x \in \{0, 1\}^\ell$ , every database  $DB$  of size  $N$  such that  $\text{Search}(f, x, DB) \neq \perp$ , and every challenge  $ch \in \mathcal{C}$ , we have

$$\Pr_{\substack{crs \leftarrow \text{Setup}(1^\lambda, N) \\ (\alpha, st_R) \leftarrow R(crs, x) \\ (\beta_1, st_S) \leftarrow S1(crs, \alpha, DB) \\ \beta_2 \leftarrow S2(crs, ch, st_S)}}} [\text{Verify}(\beta_1, ch, \beta_2, st_R) \text{ accepts}] = 1.$$

2. Weak  $\delta$ -Soundness, for every non-uniform malicious sender  $S^* = \{(S1_\lambda^*, S2_\lambda^*)\}_{\lambda \in \mathbb{N}}$ , every  $\lambda, N \in \mathbb{N}$ , and every sequence of indices  $I^* = \{i_k^*\}_{k \in [c]}$  of size  $c$ ,

$$\Pr_{\substack{crs \leftarrow \text{Setup}(1^\lambda, N) \\ x \leftarrow D \\ (\alpha, st_R) \leftarrow R(crs, x) \\ (\beta_1, st_S) \leftarrow S1^*(crs, \alpha) \\ ch \leftarrow \mathcal{C} \\ \beta_2 \leftarrow S2^*(crs, ch, st_S)}}} [\text{Verify}(\beta_1, ch, \beta_2, st_R) = I^*] \leq \delta(\lambda) + 2^{-H(\lambda)}$$

3. Receiver Privacy, for any sequence of CRS strings  $crs = \{crs_\lambda\}_{\lambda \in \mathbb{N}}$ , and for any two sequence of input strings  $x^0 = \{x_\lambda^0\}_{\lambda \in \mathbb{N}}$ ,  $x^1 = \{x_\lambda^1\}_{\lambda \in \mathbb{N}}$ ,

$$\{crs_\lambda, \alpha : (\alpha, st) \leftarrow R(crs_\lambda, x_\lambda^0)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{crs_\lambda, \alpha : (\alpha, st) \leftarrow R(crs_\lambda, x_\lambda^1)\}_{\lambda \in \mathbb{N}}.$$

4. Special Statistical Malicious Sender Privacy, there is a simulator algorithm  $\text{Sim}$ , such that, for every sequence of first message strings  $\alpha = \{\alpha_\lambda\}_{\lambda \in \mathbb{N}}$ , there exists a sequence of inputs  $x^* = \{x_\lambda^*\}$ , such that for every database  $DB$ , and for every  $ch \in \mathcal{C}$ , the following two distributions are statistically indistinguishable

- sample  $crs \leftarrow \text{Setup}(1^\lambda, N)$ , then, output  $\text{Sim}(crs, x_\lambda^*, ch, \text{Search}(f, x_\lambda^*, DB))$
- sample  $crs \leftarrow \text{Setup}(1^\lambda, N)$ , then, generate  $(\beta_1, st) \leftarrow S1(crs, \alpha_\lambda)$ , next, compute  $\beta_2 \leftarrow S2(crs, ch, st)$ , finally, output  $(\beta_1, \beta_2)$ .

5. Efficiency, both  $R$  and  $Verify$  have runtime  $\text{poly}(\lambda, \ell, c, W, \log(N))$ .

Our construction uses the following ingredients:

- A commit-and-prove system  $\Pi = (\text{Setup}, \text{FakeSetup}, \text{Com}, \text{GenFresh}, \text{P}, \text{Verify}, \text{Extract})$  for the language specified by  $f$ .

- A maliciously circuit private homomorphic encryption scheme  $\text{HE} = (\text{Enc}, \text{Eval}, \text{Dec}, \text{Sim})$  for a class of functions  $\mathcal{F} = \{\mathcal{F}_L\}_{L \in \mathbb{N}}$ .
- A somewhere statistically binding hash  $\text{SSB} = (\text{Gen}, \text{Hash}, \text{Verify}, \text{Extract})$  satisfying the properties in Definition 6.

**Construction 2 (Weakly-Sound 4-Round C-PSM).** Let  $L := L(\lambda, N)$  be a function family index such that  $\mathcal{F}_L$  includes both  $G^1$  and  $G^2$  (Figs. 2 and 3) for databases  $\text{DB}$  of size  $N$ . The construction is as follows:

- $\text{Setup}(1^\lambda, N)$ :
  - Generate a CRS for  $\Pi$ ,  $\text{crs}_\Pi \leftarrow \Pi.\text{Setup}(1^\lambda)$ .
  - Generate an SSB hash key binding to the first  $c$  indices (or any other arbitrary sequence of  $c$  indices),  $(hk, td) \leftarrow \text{SSB.Gen}(1^\lambda, N, \{i\}_{i \in [c]})$ .
  - Output  $\text{crs} := (\text{crs}_\Pi, hk)$ .
- $\text{R}(\text{crs}, x)$ :
  - Encrypt  $x$  under HE to produce  $(ct, sk) \leftarrow \text{HE.Enc}(1^\lambda, 1^L, x)$ .
  - Output  $\alpha := ct$  and store internal state  $st := sk$ .
- $\text{S1}(\text{crs}, \alpha, \text{DB})$ :
  - Parse  $\text{crs}$  and  $\alpha$  as  $(\text{crs}_\Pi, hk)$  and  $ct$  respectively.
  - Commit to every entry in  $\text{DB}$  to produce  $\widetilde{\text{DB}} = \{\widetilde{\text{DB}}_i \leftarrow \Pi.\text{Com}(\text{crs}_\Pi, \text{DB}_i; r_i^{\text{com}})\}_{i \in [N]}$ .
  - Hash  $\widetilde{\text{DB}}$  using SSB to obtain  $(h, \{\tau_i\}_{i \in [N]}) := \text{SSB.Hash}(hk, \widetilde{\text{DB}})$ .
  - Produce fresh commitments and their randomness  $\Gamma \leftarrow \Pi.\text{GenFresh}(\text{crs}_\Pi)$ .
  - Sample random coins  $r_P$  for  $\Pi.\text{P1}$ .
  - Homomorphically evaluate the function  $G^1$  on  $ct$  to obtain  $ct_{\text{eval},1} \leftarrow \text{HE.Eval}(\text{crs}_\Pi, ct, G^1_{\text{DB}, \widetilde{\text{DB}}, \{\tau_i\}_{i \in [N]}, \{r_i^{\text{com}}\}_{i \in [N]}, \Gamma, r_P})$ .
  - Output  $\beta_1 := (h, ct_{\text{eval},1})$  and store internal state  $st := (x, \text{DB}, \{r_i^{\text{com}}\}_{i \in [N]}, \Gamma, r_P)$ .
- $\text{S2}(\text{crs}, ch, st)$ :
  - Parse  $\text{crs}$  and  $st$  as  $(\text{crs}_\Pi, hk)$  and  $(x, \text{DB}, \{r_i^{\text{com}}\}_{i \in [N]}, \Gamma, r_P)$  respectively.
  - Homomorphically evaluate the function  $G^2$  on  $ct$  to obtain  $ct_{\text{eval},2} \leftarrow \text{HE.Eval}(\text{crs}_\Pi, ct, G^2_{\text{crs}_\Pi, \text{DB}, \{r_i^{\text{com}}\}_{i \in [N]}, \Gamma, r_P, ch})$ .
  - Output  $\beta_2 := ct_{\text{eval},2}$ .
- $\text{Verify}(\text{crs}, \beta_1, ch, \beta_2, st)$ :
  - Parse  $\text{crs}, \beta_1, \beta_2$  and  $st$  as  $(\text{crs}_\Pi, hk)$ ,  $(h, ct_{\text{eval},1})$ ,  $ct_{\text{eval},2}$  and  $sk$  respectively.
  - Decrypt  $ct_{\text{eval},1}$  to obtain  $(\{i_k\}_{k \in [c]}, \{\tilde{w}_k\}_{k \in [c]}, \pi_1, \{\tau_k\}_{k \in [c]}) := \text{HE.Dec}(sk, ct_{\text{eval},1})$ .
  - Decrypt  $ct_{\text{eval},2}$  to obtain  $\pi_2 := \text{HE.Dec}(sk, ct_{\text{eval},2})$ .
  - Accept and output  $\{i_k\}_{k \in [c]}$  iff  $\Pi.\text{Verify}(\text{crs}_\Pi, x, \{\tilde{w}_k\}_{k \in [c]}, ch, \pi_1, \pi_2)$  accepts and  $\forall k \in [c] : \text{SSB.Verify}(hk, h, i_k, \tilde{w}_k, \tau_k)$  accepts.

We first prove  $\delta$ -soundness and special statistical malicious sender privacy of Construction 2.

**procedure**  $G^1_{crs_{\Pi}, DB, \widetilde{DB}, \{\tau_i\}_{i \in [N]}, \Gamma, r_P}(x)$

Let  $out := \text{Search}(x, f, DB)$

**if**  $out == \perp$  **then**

Output  $\perp$

**else**

Parse  $out$  as  $out = (i_1, \dots, i_c)$ .

Generate the first prover message:

$$\pi_1 \leftarrow \text{P1}(crs_{\Pi}, x, \{DB_{i_k}\}_{k \in [c]}, \{r_{i_k}\}_{k \in [c]}, \Gamma; r_P).$$

Output  $(\{i_k\}_{k \in [c]}, \{\widetilde{DB}_{i_k}\}_{k \in [c]}, \pi_1, \{\tau_{i_k}\}_{k \in [c]})$ .

**Fig. 2.** Description of  $G^1$

**procedure**  $G^2_{crs_{\Pi}, DB, \{r_i^{com}\}_{i \in [N]}, \Gamma, r_P, ch}(x)$

Let  $out := \text{Search}(x, f, DB)$

**if**  $out == \perp$  **then**

Output  $\perp$

**else**

Parse  $out$  as  $out = (i_1, \dots, i_c)$ .

Generate the second prover message:

$$\pi_2 \leftarrow \text{P2}(crs_{\Pi}, x, \{DB_{i_k}\}_{k \in [c]}, \{r_{i_k}\}_{k \in [c]}, \Gamma, r_P, ch).$$

Output the second prover message  $\pi_2$ .

**Fig. 3.** Description of  $G^2$

**Theorem 13.** *Assuming SSB is index-hiding,  $\Pi$  has indistinguishable CRS modes, HE has semantic security, and  $\Pi$  is  $\delta$ -sound, Construction 2 is weakly  $(\delta + \gamma)$ -sound for any positive constant (or any non-negligible function)  $\gamma$ .*

*Proof.* Let  $S^* = (S1^*, S2^*)$  be a malicious sender and let  $I^* = \{i_k^*\}_{k \in [c]}$  be any sequence of indices of size  $c$ . For each hybrid  $H_j$ , define the probability  $p_j$  as follows:

$$p_j := \Pr[\Pi.\text{Verify}(crs_{\Pi}, x, \{\tilde{w}_k\}_{k \in [c]}, \pi_1, ch, \pi_2) \text{ accepts} \wedge \forall k \in [c] : \text{SSB}.\text{Verify}(hk, h, i_k^*, \tilde{w}_k, \tau_k) \text{ accepts}].$$

where in each hybrid we describe how  $crs_{\Pi}$ ,  $x$ ,  $\{\tilde{w}_k\}_{k \in [c]}$ ,  $\pi_1$ ,  $ch$ ,  $\pi_2$ ,  $hk$ ,  $h$ , and  $\{\tau_k\}_{k \in [c]}$  are defined.

**Hybrid  $H_0$ :** This is the soundness experiment. In more detail, here,

- $crs_{\Pi} \leftarrow \Pi.\text{Setup}(1^\lambda)$ ,
- $(hk, td_{SSB}) \leftarrow \text{SSB}.\text{Gen}(1^\lambda, N, \{k\}_{k \in [c]})$ ,
- $x \leftarrow D$ ,
- $(ct, sk) \leftarrow \text{HE}.\text{Enc}(1^\lambda, 1^L, x)$ ,
- $((h, ct_{eval,1}), st) \leftarrow S1^*((crs_{\Pi}, hk), ct)$ ,

- $ch \leftarrow \mathcal{C}$ ,
- $ct_{eval,2} \leftarrow S2^*(crs, ch, st)$ ,
- $(\{i_k\}_{k \in [c]}, \{\tilde{w}_k\}_{k \in [c]}, \pi_1, \{\tau_k\}_{k \in [c]}) := \text{HE.Dec}(sk, ct_{1,eval})$ ,
- and  $\pi_2 := \text{HE.Dec}(sk, ct_{eval,2})$ .

**Hybrid  $H_1$ :** This is identical to  $H_0$  except that here  $hk$  is generated binding to indices  $i_1^*, \dots, i_c^*$ , i.e.,  $(hk, td_{ssb}) \leftarrow \text{SSB.Gen}(1^\lambda, N, \{i_k^*\}_{k \in [c]})$ . The index hiding property of SSB implies that  $H_0 \stackrel{c}{\approx} H_1$ . Consequently,  $|p_0 - p_1| = \text{negl}(\lambda)$ .

**Hybrid  $H_2$ :** The only difference between this hybrid and  $H_1$  is that here,  $crs_\Pi$  is generated along with a trapdoor  $td_\Pi$  via  $(crs_\Pi, td_\Pi) \leftarrow \Pi.\text{FakeSetup}(1^\lambda)$ . Since  $\Pi$  has indistinguishable CRS modes,  $H_1 \stackrel{c}{\approx} H_2$ . Therefore,  $|p_1 - p_2| = \text{negl}(\lambda)$ .

**Lemma 1.** *Assuming HE is semantically secure,  $p_2 - (\delta + 2^{-H}) = \text{negl}(\lambda)$ .*

*Proof.* Using  $S^*$  we build an adversary  $\mathcal{A}$  against the semantic security of HE.  $\mathcal{A}$  works as follows:

- It generates  $crs_\Pi$ ,  $hk$ , and  $td_{ssb}$  exactly as in  $H_2$ .
- It samples two elements  $x_0 \leftarrow D, x_1 \leftarrow D$ .
- $\mathcal{A}$  sends  $x_0, x_1$  to the semantic security challenger of HE.
- It receives as response an HE ciphertext  $ct$  from the HE semantic security challenger. The ciphertext  $ct$  either encrypts  $x_0$  or  $x_1$  under an honestly generated HE key  $sk$ .
- $\mathcal{A}$  runs  $S1^*$  to obtain  $((h, ct_{eval,1}), st) \leftarrow S1^*((crs_\Pi, hk), ct)$
- $\mathcal{A}$  receives a random challenge  $ch \leftarrow \mathcal{C}$ .
- $\mathcal{A}$  runs  $S2^*$  to obtain  $ct_{eval,2} \leftarrow S2^*((crs_\Pi, hk), st)$ .
- Using  $td_{ssb}$  it recovers commitments  $\{\tilde{w}_k^*\}_{k \in [c]} := \text{SSB.Extract}(td_{ssb}, h)$ . Using  $td_{com}$ , for each  $k \in [c]$  it recovers  $w_k^* := \text{Com.Extract}(td_{com}, \tilde{w}_k^*)$ .
- If  $f(x_0, \{w_k^*\}_{k \in [c]}) = 1$ , it outputs 1. Otherwise, it outputs 0.

Now we analyze the success probability of  $\mathcal{A}$  in breaking the semantic security of HE. Let

$$(\{i_k\}_{k \in [c]}, \{\tilde{w}_k\}_{k \in [c]}, \pi_1, \{\tau_k\}_{k \in [c]}) := \text{HE.Dec}(sk, ct_{eval,1}).$$

First, we consider the case where  $ct$  encrypts  $x_0$ . In this case with probability at least  $p_2$ ,

$$\forall k \in [c] : \text{SSB.Verify}(hk, h, i_k^*, \tilde{w}_k, \tau_k) \text{ accepts,} \quad (2)$$

and

$$\Pi.\text{Verify}(crs_\Pi, x_0, \{\tilde{w}_k\}_{k \in [c]}, \pi_1, ch, \pi_2) \text{ accepts.} \quad (3)$$

By extractability of SSB, the former implies that  $\forall k \in [c] : \tilde{w}_k = \tilde{w}_k^*$ . Consequently, by  $\delta$ -soundness of  $\Pi$ , with probability at least  $p_2 - \delta$ ,  $f(x_0, \{w_k^*\}_{k \in [c]}) = 1$ . We conclude that in this case  $\mathcal{A}$  outputs 1 with probability at least  $p_2 - \delta$ . Now we turn to the other case where  $ct$  encrypts  $x_1$ . In this case,  $x_0$  maintains all of its entropy, therefore, since  $f$  is  $H$ -instance entropic,

$$\Pr[f(x_0, \{w_k^*\}_{k \in [c]}) = 1] = 2^{-H},$$

i.e.,  $\mathcal{A}$  outputs 1 with probability  $2^{-H}$ . We showed that  $\mathcal{A}$  breaks the semantic security of HE with probability at least  $p_2 - \delta - 2^{-H}$ .

This concludes the proof.

**Theorem 14.** *Assuming HE is maliciously circuit private,  $\Pi$  satisfies special statistical zero-knowledge, and  $\Pi$  has statistically hiding commitments, Construction 2 satisfies special statistical malicious sender privacy.*

*Proof.* Let  $\alpha$  be an arbitrary first message,  $ch \in \mathcal{C}$  be any challenge, DB be any database of size  $N \in \mathbb{N}$ , and let  $crs \leftarrow \text{Setup}(1^\lambda, N)$  be a  $crs$  generated through Setup. First, we describe the simulator algorithm Sim.

- Sim receives as input a CRS parsed as  $crs := (crs_\Pi, hk)$ , a first message  $\alpha := ct$ , a bitstring  $x^*$ , and indices  $\{i_k^*\}_{k \in [c]}$  (W.L.O.G assume that the indices are not  $\perp$ ).
- Using the zero-knowledge simulator for  $\Pi$ , it computes  $(\{\tilde{w}_k^*\}_{k \in [c]}, \pi_1^*, \pi_2^*) \leftarrow \Pi.\text{Sim}(crs_\Pi, x, ch)$ .
- For each  $i \in [N]/\{i_k^*\}_{k \in [c]}$ , Sim computes a commitment  $\tilde{\text{DB}}_i \leftarrow \Pi.\text{Commit}(crs_{com}, \mathbf{0})$ . For each  $k \in [c]$  it sets the  $i_k^*$ th commitment to be equal to  $\tilde{\text{DB}}_{i_k^*} := \tilde{w}_k^*$ .
- It hashes  $\tilde{\text{DB}}$  to obtain  $(h, \{\tau_i\}_{i \in [N]}) := \text{SSB.Hash}(hk, \tilde{\text{DB}})$ .
- Using the HE simulator it computes

$$ct_{eval,1} \leftarrow \text{HE.Sim}(ct, (\{i_k^*\}_{k \in [c]}, \{\tilde{w}_k^*\}_{k \in [c]}, \pi_1^*, \{\tau_{i_k^*}\}_{k \in [c]})).$$

- Using the HE simulator it computes

$$ct_{eval,2} \leftarrow \text{HE.Sim}(ct, \pi_2^*)$$

- It outputs  $(h, ct_{eval,1}, ct_{eval,2})$ .

We now proceed via a series of hybrids to show that the output of Sim is statistically indistinguishable from an honestly generated sender message.

**Hybrid  $H_0$ :** This hybrid corresponds to generating the sender messages  $\beta_1, \beta_2$  honestly through  $(\beta_1, st) := (h, ct_{eval}) \leftarrow \text{S1}(crs, \alpha, \text{DB})$  and  $\beta_2 := ct_{eval} \leftarrow \text{S2}(crs, ch, st)$ .

**Hybrid  $H_1$ :** This hybrid uses HE.Sim to produce  $ct_{eval,1}$  and  $ct_{eval,2}$ . In more detail, given  $ct$ , we know that there exists an  $x^*$  such that,

$$\text{HE.Eval}(ct, G_{crs_\Pi, \text{DB}, \tilde{\text{DB}}, \{\tau_i\}_{i \in [N]}, \Gamma, r_P}^1) \stackrel{\approx}{\approx} \text{HE.Sim}(ct, G_{crs_\Pi, \text{DB}, \tilde{\text{DB}}, \{\tau_i\}_{i \in [N]}, \Gamma, r_P}^1(x^*)),$$

and

$$\text{HE.Eval}(ct, G_{crs_\Pi, \text{DB}, \{r_i^{com}\}_{i \in [N]}, \Gamma, r_P, ch}^2) \stackrel{\approx}{\approx} \text{HE.Sim}(ct, G_{crs_\Pi, \text{DB}, \{r_i^{com}\}_{i \in [N]}, \Gamma, r_P, ch}^2(x^*)).$$

In this hybrid,  $ct_{eval,1}$  and  $ct_{eval,2}$  are generated as

$$ct_{eval,1} \leftarrow \text{HE.Sim}(ct, G_{crs_\Pi, \text{DB}, \tilde{\text{DB}}, \{\tau_i\}_{i \in [N]}, \Gamma, r_P}^1(x^*)),$$

and

$$ct_{eval,2} \leftarrow \text{HE.Sim}(ct, G_{crs_\Pi, \text{DB}, \{r_i^{com}\}_{i \in [N]}, \Gamma, r_P, ch}^2(x^*)).$$

It follows from the malicious circuit privacy of HE that  $H_0 \stackrel{s}{\approx} H_1$ .

**Hybrid  $H_2$ :** The difference between this hybrid and the previous hybrid is only syntactical. In this hybrid, to generate  $ct_{eval,1}$  and  $ct_{eval,2}$ , first, the (lexicographically) smallest indices  $\{i_k^*\}_{k \in [c]}$  such that  $f(x^*, \{\text{DB}_{i_k^*}\}_{k \in [c]}) = 1$  are computed. Next,  $\pi_1$  and  $\pi_2$  are computed as

$$\pi_1 \leftarrow P1(crs_{\Pi}, x, \{\text{DB}_{i_k^*}\}_{k \in [c]}, \{r_{i_k^*}\}_{k \in [c]}, \Gamma; r_P)$$

and

$$\pi_2 \leftarrow P2(crs_{\Pi}, x, \{\text{DB}_{i_k^*}\}_{k \in [c]}, \{r_{i_k^*}\}_{k \in [c]}, \Gamma, r_P, ch).$$

Finally,  $ct_{eval,1}$  and  $ct_{eval,2}$  are computed as

$$ct_{eval,1} \leftarrow \text{HE.Sim}(ct, (\{i_k^*\}_{k \in [c]}, \{\widetilde{\text{DB}}_{i_k^*}\}_{k \in [c]}, \pi_1, \{\tau_{i_k^*}\}_{k \in [c]}),$$

and

$$ct_{eval,2} \leftarrow \text{HE.Sim}(ct, \pi_2).$$

As already stated  $H_1$  and  $H_2$  are identical.

**Hybrid  $H_3$ :** In this hybrid we modify how  $\widetilde{\text{DB}}$  is generated. Here, for each  $k \in [c]$ ,

$$\widetilde{\text{DB}}_{i_k^*} \leftarrow \Pi.\text{Commit}(crs_{\Pi}, \text{DB}_{i_k^*}; r_{i_k^*}^{com})$$

as before, but the rest of the commitments are generated as

$$\{\widetilde{\text{DB}}_i \leftarrow \Pi.\text{Commit}(crs_{\Pi}, \mathbf{0})\}_{i \in [N]/\{i_k^*\}_{k \in [c]}}.$$

Notice that we don't modify the commitments whose randomness are used in the HE.Sim algorithm. Therefore, by the statistical hiding property of the commitments in  $\Pi$ ,  $H_2 \stackrel{s}{\approx} H_3$ .

**Hybrid  $H_4$ :** The difference between this hybrid and the previous hybrid is that here  $\{\widetilde{\text{DB}}_{i_k^*}\}_{k \in [c]}$ ,  $\pi_1$ , and  $\pi_2$  are generated using the simulator for  $\Pi$ , i.e.,

$$(\{\widetilde{\text{DB}}_{i_k^*}\}_{k \in [c]}, \pi_1, \pi_2) \leftarrow \Pi.\text{Sim}(x^*, ch).$$

The special zero-knowledge property of  $\Pi$  directly implies that  $H_3 \stackrel{s}{\approx} H_4$ . Observe that,  $H_4$  corresponds to generating the sender messages via Sim.

Depending on how HE is instantiated, Construction 2 can support different classes of predicates with different trade-offs in terms of black-box usage of underlying cryptographic primitives. If we instantiate HE with Theorem 8, we can have a black-box construction supporting NC<sup>1</sup> predicates  $f$  where  $\text{Search}(\cdot, f, \text{DB})$  can be implemented in by a branching program whose length is logarithmic in  $|\text{DB}|$ .

**Theorem 15.** *Assuming hardness of either of DDH or LWE, there exists a family of weakly-sound 4-round C-PSM protocols with the following properties:*

1. It supports all predicates  $f$  such that  $f$  can be implemented by an  $\text{NC}^1$  circuit and also for every database  $\text{DB}$  of size  $N$ ,  $\text{Search}(\cdot, f, \text{DB})$  can be implemented by a branching program of length logarithmic in  $N$ .
2. It only makes black-box use of the underlying cryptographic primitives.
3. It is receiver private.
4. It is weakly  $\delta$ -sound.
5. It satisfies special statistical malicious sender privacy.

*Proof.* We instantiate Construction 2 with the black-box maliciously circuit private homomorphic encryption scheme of Theorem 8 for the class of branching programs  $\{\mathcal{B}_L\}_{L \in \mathbb{N}}$ . We have already proven weak  $\delta$ -soundness and special statistical malicious sender privacy of Construction 2. Correctness follows from the correctness of HE, correctness of  $\Pi$ , and correctness of SSB. Receiver privacy follows from the semantic security of HE. For efficiency, we need to show that both  $G^1$  and  $G^2$  can be evaluated by a branching program of length  $L = \text{poly}(\lambda, \log N)$ . Observe that both  $G^1$  and  $G^2$  access the whole database only through the Search functionality. Therefore, since the Search functionality for  $f$  can be implemented by a branching program of length logarithmic in  $N$ ,  $L$  is also logarithmic in  $N$ . Furthermore, since  $f$  is in  $\text{NC}^1$ , by Theorem 7, both P1 and P2 are also in  $\text{NC}^1$ . Consequently, by Barrington's theorem [3], P1 and P2 can be implemented by a polynomial (in  $\lambda$ ) length branching program. Therefore,  $L = \text{poly}(\lambda, \log N)$ .

Alternatively, we can instantiate HE with Theorem 9 to get a construction supporting all bounded depth circuits. While this construction only makes black-box use of HE, however, the homomorphic encryption scheme constructed in Theorem 9 is non-black-box due to relying on bootstrapping.

**Theorem 16.** *Assuming hardness of LWE, there exists a family of weakly-sound 4-round C-PSM protocols with the following properties:*

1. It supports all predicates  $f$  such that  $f$  can be implemented by bounded-depth circuits, i.e., the C-PSM protocol is leveled.
2. Its only non-black-box use of the underlying cryptographic primitives happens through bootstrapping.
3. It is receiver private.
4. It is weakly  $\delta$ -sound.
5. It satisfies special statistical malicious sender privacy.

*Proof.* We instantiate Construction 2 with the maliciously circuit private homomorphic encryption scheme of Theorem 9 for the class of circuits  $\{\mathcal{F}_L\}_{L \in \mathbb{N}}$ , where for each  $L \in \mathbb{N}$ ,  $\mathcal{F}_L$  consists of all circuits of depth at most  $L$ . Establishing weak  $\delta$ -soundness, special statistical malicious sender privacy, correctness and receiver privacy is identical to Theorem 15. For efficiency, it is straightforward to verify that  $G^1$  and  $G^2$  can be evaluated by circuits of depth  $L = \text{poly}(\lambda, \log N)$ .

## 5.2 4-Round to 2-Round Transformation

Here we provide a generic transformation that converts any weakly-sound 4-round C-PSM protocol to a weakly-sound 2-round protocol. Analogously to weakly-sound 4-round C-PSM, we define weakly-sound 2-round C-PSM as follows:

**Definition 11 (Weakly Sound 2-Round C-PSM).** *Let  $\ell, c, W, f, H, D$  be the same as Definition 9. A weakly sound C-PSM for  $f$ , is a protocol between a sender and a receiver described by a tuple of PPT algorithms  $(\text{Setup}, R, S, \text{Verify})$ , where the interface of  $\text{Setup}, R$  and  $S$  is identical to their interface in Definition 9 and  $\text{Verify}$  has the following syntax:*

- $\text{Verify}(\beta, st)$ , on input a sender message  $\beta$  and internal state  $st$ , either accepts and outputs a sequence  $I = \{i_k\}_{k \in [c]}$  of indices, or rejects.

Except for  $\delta$ -soundness we require the protocol to satisfy all properties in Definition 9. Additionally, we consider the following weaker variant of soundness:

1. Weak  $\delta$ -Soundness, for every non-uniform malicious sender  $S^* = \{S_\lambda^*\}_{\lambda \in \mathbb{N}}$ , every  $\lambda, N \in \mathbb{N}$ , and every sequence of indices  $I^* = \{i_k^*\}_{k \in [c]}$  of size  $c$ ,

$$\Pr_{\substack{crs \leftarrow \text{Setup}(1^\lambda, N) \\ x \leftarrow D \\ (\alpha, st) \leftarrow R(crs, x) \\ \beta \leftarrow S^*(crs, \alpha)}}} [\text{Verify}(\beta, st) = I^*] \leq \delta(\lambda) + 2^{-H(\lambda)}$$

Our transformation uses the following ingredients:

- A 4-round weakly sound C-PSM protocol  $\Sigma = (\text{Setup}, R, S1, S2, \text{Verify})$ .
- A dual-mode statistically sender private OT scheme  $\text{OT} = (\text{Setup}, \text{FakeSetup}, \text{Extract}, \text{OT1}, \text{OT2}, \text{OT3})$ .

**Construction 3.** The construction is as follows:

- $\text{Setup}(1^\lambda, N)$ :
  - Generate a CRS for  $\Sigma$ ,  $crs_\Sigma \leftarrow \Sigma.\text{Setup}(1^\lambda, N)$ .
  - Generate a CRS for dual-mode OT,  $crs_{OT} \leftarrow \text{OT}.\text{Setup}(1^\lambda)$ .
  - Output  $crs := (crs_\Sigma, crs_{OT})$ .
- $R(crs, x)$ :
  - Generate a  $\Sigma$  first message for  $x$  along with an internal state,  $(\alpha_\Sigma, st_\Sigma) \leftarrow \Sigma.R(crs_\Sigma, x)$ .
  - Sample a random challenge  $ch \leftarrow \mathcal{C}$  from the challenge space of  $\Sigma$ .
  - Generate an OT first message for  $ch$  along with an internal state,  $(ot_1, st_{OT}) \leftarrow \text{OT}.\text{OT1}(crs_{OT}, ch)$ .
  - Output first message  $\alpha := (\alpha_\Sigma, ot_1)$  and internal state  $st = (x, ch, st_\Sigma, st_{OT})$ .
- $S(crs, \alpha, \text{DB})$ :
  - Parse  $crs$  and  $\alpha$  as  $(crs_\Sigma, crs_{OT})$  and  $(\alpha_\Sigma, ot_1)$  respectively.
  - Compute  $(\beta_1, st) \leftarrow \Sigma.S1(crs_\Sigma, \alpha_\Sigma, \text{DB})$ .



- For each  $ch \in \mathcal{C}$  compute  $\beta_{2,ch} \leftarrow \Sigma.S2(crs_\Sigma, st, ch, \text{DB})$ .
  - Compute OT second message  $ot_2 \leftarrow \text{OT.OT2}(ot_1, \{\beta_{2,ch}\}_{ch \in \mathcal{C}})$ .
  - Output  $\beta := (\beta_1, ot_2)$ .
- **Verify**( $\beta, st$ ):
- Parse  $\beta$  and  $st$  as  $(\beta_1, ot_2)$  and  $(x, ch, st_\Sigma, st_{OT})$  respectively.
  - Recover  $\beta_{2,ch}$  as  $\beta_{2,ch} := \text{OT.OT3}(ot_2, st_{OT})$ .
  - Output whatever  $\Sigma.\text{Verify}(\beta_1, ch, \beta_{2,ch}, st_\Sigma)$  outputs.

The correctness immediately follows from the correctness of  $\Sigma$  and OT. If the size of the challenge space of  $\mathcal{C}$  is a constant (or scales logarithmically with  $N$ ) then, the efficiency also directly follows from the efficiency of  $\Sigma$ . In the full version of this paper we prove the following two theorems.

**Theorem 17 (Weak  $\delta$ -Soundness).** *Assuming  $\Sigma$  satisfies weak  $\delta$ -soundness, Construction 3 satisfies weak  $(\delta + \gamma)$ -soundness for every constant (or even non-negligible function)  $\gamma > 0$ .*

**Theorem 18 (Statistical Malicious Sender Privacy).** *Assuming OT is statistical sender private, and  $\Sigma$  satisfies special statistical malicious sender privacy, Construction 3 is statistically malicious circuit private.*

### 5.3 Weakly $\delta$ -Sound to $\text{negl}(\lambda)$ -Sound Transformation

Here we present a generic transformation that for any constant  $\delta > 0$  converts a weakly  $\delta$ -sound 2-round C-PSM to a  $\text{negl}(\lambda)$ -sound 2-round C-PSM. The transformation is essentially parallel repetition of the weakly-sound protocol, but the verification algorithm also checks that all the repetitions return the same set of indices.

For the following construction, let  $\Sigma = (\text{Setup}, \text{R}, \text{S}, \text{Verify})$  be any weakly sound 2-round C-PSM with  $\delta$ -soundness.

**Construction 4.** Let  $rep := rep(\lambda, N, c)$  be a parameter indicating the number of repetitions. The construction is as follows:

- **Setup**( $1^\lambda, N$ ):
  - Generate and output  $rep$  independent CRSs for  $\Sigma$ ,  $crs := \{crs_\Sigma^i \leftarrow \Sigma.\text{Setup}(1^\lambda, N)\}_{i \in [rep]}$ .
- **R**( $crs, x$ ):
  - Generate  $rep$  first messages for  $\Sigma$  along with their internal state,  $\{(\alpha_\Sigma^i, st_\Sigma^i) \leftarrow \Sigma.\text{R}(crs_\Sigma^i, x)\}_{i \in [rep]}$ .
  - Output the first messages  $\alpha := \{\alpha_\Sigma^i\}_{i \in [rep]}$  and internal state  $st = (x, \{st_\Sigma^i\}_{i \in [rep]})$ .
- **S**( $crs, \alpha, \text{DB}$ ):
  - Compute and output  $rep$  second messages for  $\Sigma$ ,  $\beta := \{\beta_\Sigma^i \leftarrow \Sigma.\text{S}(crs_\Sigma^i, \alpha_\Sigma^i, \text{DB})\}_{i \in [rep]}$ .
- **Verify**( $\beta, st$ ):
  - Accept iff each repetition accepts and outputs a sequence of indices of size  $c$   $\{I_i := \Sigma.\text{Verify}\beta_i, st_i\}_{i \in [rep]}$  and all the sequences  $I_i$  are equal.

The correctness and statistical malicious sender privacy of Construction 4 immediately follow because the same properties hold in  $\Sigma$ . This construction satisfies efficiency as long as  $rep$  grows at most logarithmically in  $N$ .

**Theorem 19.** *If  $\Sigma$  is weakly  $\delta$ -sound, then, Construction 4 is  $N^c \cdot \delta^{rep}$ -sound.*

*Proof.* For each possible sequence  $I^*$ , the probability that all of the repetitions accept and output  $I^*$  is at most  $\delta^{rep}$ . Since we have at most  $N^c$  different sequences, the theorem follows.

By setting  $rep := (\lambda + c \cdot \log(N)) / \log(1/\delta)$  we get  $2^{-\lambda}$  soundness.

## 5.4 Putting Everything Together

In this section, we combine the constructions in Subject. 5.1 with the transformations in Subject. 5.2 and Subject. 5.3, to obtain 2-round C-PSM constructions for richer classes of functionalities.

**Theorem 20.** *Assuming hardness of either of DDH or LWE, there exists a family of 2-round C-PSM protocols in the CRS model with the following properties:*

1. *It supports all predicates  $f$  such that  $f$  can be implemented by an  $NC^1$  circuit and also for every database  $DB$  of size  $N$ ,  $\text{Search}(\cdot, f, DB)$  can be implemented by a branching program of length logarithmic in  $N$ .*
2. *It only makes black-box use of the underlying cryptographic primitives.*
3. *It is receiver private.*
4. *It is (strongly) sound.*
5. *It satisfies statistical malicious sender privacy.*
6. *It has transparent setup, i.e., the CRS is simply a random string.*

**Theorem 21.** *Assuming hardness of LWE, there exists a family of 2-round C-PSM protocols in the CRS model with the following properties:*

1. *It supports all predicates  $f$  such that  $f$  can be implemented by bounded-depth circuits, i.e., the C-PSM protocol is leveled.*
2. *Its only non-black-box use of the underlying cryptographic primitives happens through bootstrapping.*
3. *It is receiver private.*
4. *It is (strongly) sound.*
5. *It satisfies statistical malicious sender privacy.*
6. *It has transparent setup.*

**Acknowledgments.** Sanjam Garg is supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. Omkant Pandey is supported in part by DARPA SIEVE Award HR00112020026, NSF CAREER Award 2144303, NSF grants 2028920, 2106263, and 2128187. 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, Sloan Foundation, Visa Inc., or NSF.

## References

1. Aggarwal, D., Döttling, N., Dujmovic, J., Hajiabadi, M., Malavolta, G., Obremski, M.: Algebraic restriction codes and their applications. In: ITC, pp. 2:1–2:15 (2022)
2. Apple Inc: Password monitoring - apple support (2021). <https://support.apple.com/guide/security/password-monitoring-sec78e79fc3b/web>
3. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in  $NC^1$ . In: STOC, pp. 1–5 (1986)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
5. Chase, M., Garg, S., Hajiabadi, M., Li, J., Miao, P.: Amortizing rate-1 OT and applications to PIR and PSI. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13044, pp. 126–156. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-90456-2\\_5](https://doi.org/10.1007/978-3-030-90456-2_5)
6. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: CCS, pp. 1223–1237 (2018)
7. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: CCS, pp. 1243–1255 (2017)
8. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* **45**(6), 965–981 (1998)
9. Cong, K., et al.: Labeled PSI from homomorphic encryption with reduced computation and communication. In: CCS, pp. 1135–1150 (2021)
10. Döttling, N., Dujmovic, J.: Maliciously circuit-private FHE from information-theoretic principles. In: ITC (2022)
11. Google Inc: Protect your accounts from data breaches with password checkup (2019). <https://security.googleblog.com/2019/02/protect-your-accounts-from-data.html>
12. Hubáček, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: ITCS, pp. 163–172 (2015)
13. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC, pp. 21–30 (2007)
14. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-70936-7\\_31](https://doi.org/10.1007/978-3-540-70936-7_31)
15. Izabachène, M., Nitulescu, A., de Perthuis, P., Pointcheval, D.: Myope: malicious security for oblivious polynomial evaluation. In: SCN, pp. 663–686 (2022)
16. Kannepalli, S., Laine, K., Moreno, R.C.: Password monitor: Safeguarding passwords in microsoft edge (2021). <https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/>
17. MarlinSpike, M.: The difficulty of private contact discovery (2014). <https://whispersystems.org/blog/contact-discovery/>
18. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988). [https://doi.org/10.1007/3-540-48184-2\\_32](https://doi.org/10.1007/3-540-48184-2_32)
19. Micali, S., Rabin, M.O., Kilian, J.: Zero-knowledge sets. In: FOCS, pp. 80–91 (2003)
20. 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. LNCS, vol. 9452, pp. 121–145. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48797-6\\_6](https://doi.org/10.1007/978-3-662-48797-6_6)

21. Ostrovsky, R., Paskin-Cherniavsky, A., Paskin-Cherniavsky, B.: Maliciously circuit-private FHE. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 536–553. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_30](https://doi.org/10.1007/978-3-662-44371-2_30)
22. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85174-5\\_31](https://doi.org/10.1007/978-3-540-85174-5_31)



# Improved Private Set Intersection for Sets with Small Entries

Dung Bui<sup>1</sup> and Geoffroy Couteau<sup>2</sup>(✉)

<sup>1</sup> IRIF, Université Paris Cité, Paris, France

bui@irif.fr

<sup>2</sup> CNRS, IRIF, Université Paris Cité, Paris, France

couteau@irif.fr

**Abstract.** We introduce new protocols for private set intersection (PSI), building upon recent constructions of pseudorandom correlation generators, such as vector-OLE and ring-OLE. Our new constructions improve over the state of the art on several aspects, and perform especially well in the setting where the parties have databases with small entries. We obtain three main contributions:

1. We introduce a new semi-honest PSI protocol that combines subfield vector-OLE with hash-based PSI. Our protocol is the first PSI protocol to achieve communication complexity *independent* of the computational security parameter  $\kappa$ , and has communication lower than all previous known protocols for input sizes  $\ell$  below 70 bits.
2. We enhance the security of our protocol to the malicious setting, using two different approaches. In particular, we show that applying the *dual execution technique* yields a malicious PSI whose communication remains independent of  $\kappa$ , and improves over all known PSI protocols for small values of  $\ell$ .
3. As most previous protocols, our above protocols are in the random oracle model. We introduce a third protocol which relies on subfield ring-OLE to achieve maliciously secure PSI in the *standard model*, under the ring-LPN assumption. Our protocol enjoys extremely low communication, reasonable computation, and standard model security. Furthermore, it is *batchable*: the message of a client can be reused to compute the intersection of their set with that of multiple servers, yielding further reduction in the overall amortized communication.

## 1 Introduction

*Private Set Intersection (PSI)* is a cryptographic primitive that allows parties to jointly compute the set of all common elements between their datasets, without leaking any value outside of the intersection. It is a special case of secure multi-party computation (MPC). PSI enjoys a wide array of real-life applications; it is perhaps the most actively researched concrete functionality in secure computation, and has been the target of a tremendous number of works, see [10, 13, 18–24, 26–28] and references therein for a sample. As a consequence of this intense research effort, modern PSI protocols now achieve impressive efficiency features,

communicating only a few hundred bits per database items, and processing millions of items in seconds.

**Improving PSI with Pseudorandom Correlation Generators.** Pseudorandom correlation generators (PCG) have been introduced in the works of [3, 5, 8] and have been the subject of a long and fruitful line of work [3–8, 11, 30, 32, 34]. At a high level, a PCG allows two parties to securely stretch long pseudorandom correlated strings from short, correlated seeds. Securely sharing correlated random strings is a crucial component in most modern secure computation protocols, which operate in the preprocessing model; PCG allows to realize this functionality with almost no communication. Among their many applications, PCGs allow to construct *silent oblivious transfer extension* protocols [4], which can realize (pseudorandom) OT extension with minimal (logarithmic) communication.

Since the top-performing PSI protocols rely on efficient OT extension, using PCG-based techniques to improve their efficiency is a natural idea. And indeed, this was done recently for OKVS-based PSI in [27], leading to the most efficient PSI protocol known to date (OKVS stands for oblivious key-value store [13]; the use of OKVS is the leading paradigm for the design of PSI protocols). To give a single datapoint, computing the intersection between two databases of size  $n = 2^{20}$  with the protocol of [27] communicates as little as  $426n$  bits in total. In addition, some of the tools used in [27] have been significantly improved since: replacing their OKVS (which is the PaXoS OKVS of [21]) by the more recent 3H-GCT OKVS of [13], and replacing their PCG (which is the one from [32]) by the recent PCG of [11], the cost goes down to an impressive  $247n$  bits of total communication. In comparison, even the *insecure* approach of exchanging the hashes of all items in the databases already requires  $160n$  bits of communication. OKVS-based PSI protocols are now firmly established as the leading paradigm in the field, and the use of PCGs to reduce their communication overhead even more seems to further widen the gap with the other paradigms.

## 1.1 Our Contributions

We thoroughly investigate how the use pseudorandom correlation generators can reduce communication in PSI protocols. We obtain several contributions:

- A new family of semi-honest hash-based PSI protocols. Our protocols can be instantiated using several hashing techniques, and achieve very low communication, especially for databases whose entries have a small bitlength.
- New maliciously secure hash-based PSI protocols. Here, interestingly, we revive the dual execution technique, which had been used previously to design malicious PSI protocols in [26], but was considered outdated. We show that, combined with our new approach, it leads to very competitive protocols, which achieve lower communication than all known alternatives for databases with small entries.
- Eventually, we design a new maliciously secure polynomial-based PSI protocol. Our protocol enjoys several powerful features: competitive communication, security in the standard model under the ring-LPN assumption (in contrast, other maliciously secure PSI use the ROM), and the possibility for

a client to publish a single encoding of its database, and later retrieve the intersection of its database with that of multiple servers independently, with a single server-to-client message, plus minimal (database-independent) additional communication.

Below, we elaborate on each of our contributions.

**Low Communication PSI for Databases with Small Entries.** Modern PSI protocols have communication  $O(\kappa \cdot n)$ , where  $n$  is the database size, and  $\kappa$  is a computational security parameter. More precisely, the receiver-to-sender communication is  $O(\kappa n)$ , while the sender-to-receiver communication is  $O(\lambda \cdot n)$ , where  $\lambda$  is a *statistical* security parameter (typically,  $\kappa = 128$  and  $\lambda = 40$ ). We introduce a new protocol, that combines hashing techniques (e.g. Cuckoo hashing or its variants, as initially used in [18]) with a new PCG-based oblivious pseudorandom function (OPRF). In contrast to all previous works, our work avoids the  $O(\kappa \cdot n)$  overhead: it reduces the receiver-to-sender communication to be roughly  $\ell \cdot n$  (where  $\ell$  is the bitsize of the database items), leading to a significant reduction in the overall communication. To our knowledge, our protocol is the first to achieve communication independent of  $\kappa$  (up to low order terms). To give a datapoint, for  $n = 2^{20}$ , with 64-bit entries, our protocol communicates  $210n$  bits, and with 32-bit entries, it communicates only  $148n$  bits. For the same parameters, the leading OKVS-based PSI of [27] communicates  $197n$  bits, even after improving it with all relevant optimization (such as using the 3H-GCT OKVS of [13], and the recent PCG of [11]). We provide further datapoints and comparisons to the state of the art on Table 1, when instantiating our protocols with various hashing methods.

**Fast Maliciously-Secure PSI for Small Entries.** We then turn our attention to maliciously secure PSI. We provide two alternative protocols which achieve malicious security; both use standard paradigms for upgrading PSI to malicious security. The first protocol combines our new PCG-based OPRF with simple hashing, and applies the standard paradigm used in most previous OKVS-based PSI to achieve malicious security (e.g. [27]). This requires to increase the sender-to-receiver message length, from  $O(\lambda \cdot n)$  to  $O(\kappa \cdot n)$  ( $\lambda$  is a statistical security parameter,  $\kappa$  is a computational security parameter; typically,  $\lambda = 40$  and  $\kappa = 128$ ) to allow for extraction of the sender input. Along the way, we also notice a small mistake in the parameter choices of [27]: they devise a new ROM-based extraction strategy in the malicious setting, and prove that a  $Q$ -query adversary will make extraction fail with probability bounded  $Q \cdot n / 2^\kappa$  (this is the probability that one of the  $Q$  queries of the malicious receiver collides with an element of the sender set). This implies that, to target 128 bits of computational security, one must set  $\kappa = 128 + \log n$ . However, the numbers reported in [27] correspond to choosing  $\kappa = 128$  at the 128-bit security level. We took this minor inconsistency into account in our tables.

More interestingly, our second protocol applies *dual execution* [26] to our PCG-based protocol with simple hashing. We observe that, in our context, this

**Table 1.** Comparison of the communication cost of several PSI protocols in the semi-honest setting and in the malicious setting, for various choices of the database size  $n$  (we assume that both parties have a database of the same size).  $\ell$  denote the bit-length of the inputs in the database; we set the computational security parameter  $\kappa$  to 128 and the statistical security parameter  $\lambda$  to 40 (for usual applications) or 30 (which can be suitable for lower risk applications). For all protocols, we take into account the optimization of [31] which reduces the costs of sending  $n$  elements of bitlength  $\lambda + 2 \cdot \log n$  to  $n \cdot (\lambda + \log n)$ . GCH stands for Generalized Cuckoo hashing (here, with 2 hash functions and 3 items per bin), 2CH for 2-choice hashing, and SH for simple hashing ( $N$  is the number of bins).

	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{20}$	$n = 2^{24}$
<b>Semi-honest setting</b>				
KKRT16 [18]	$930n$	$936n$	$948n$	$960n$
PRTY19 [20] low*	$491n$	$493n$	$493n$	$494n$
PRTY19 [20] fast*	$560n$	$571n$	$579n$	$587n$
CM20 [10]	$668n$	$662n$	$674n$	$676n$
PRTY20 [21]	$1244n$	$1192n$	$1248n$	$1278n$
RS21 [27]	$2024n$	$898n$	$406n$	$374n$
RS21 [27] enhanced**	$280n$	$260n$	$263n$	$275n$
Ours ( $\ell = 64$ , GCH)	$246n$	$220n$	$210n$	$209n$
Ours ( $\ell = 48$ , GCH)	$215n$	$189n$	$179n$	$178n$
Ours ( $\ell = 32$ , GCH)	$184n$	$158n$	$148n$	$147n$
Ours ( $\ell = 64$ , 2CH)	$214n$	$190n$	$183n$	$185n$
Ours ( $\ell = 48$ , 2CH)	$193n$	$169n$	$162n$	$164n$
Ours ( $\ell = 32$ , 2CH)	$171n$	$148n$	$141n$	$142n$
Ours ( $\ell = 64$ , SH, $N = n/10$ )	$332n$	$302n$	$284n$	$276n$
Ours ( $\ell = 48$ , SH, $N = n/10$ )	$261n$	$230n$	$209n$	$198n$
Ours ( $\ell = 32$ , SH, $N = n/10$ )	$191n$	$158n$	$133n$	$120n$
Ours ( $\ell = 64$ , SH, $N = 1$ ) ***	$154n$	$131n$	$125n$	$128n$
Ours ( $\ell = 48$ , SH, $N = 1$ ) ***	$138n$	$115n$	$109n$	$112n$
Ours ( $\ell = 32$ , SH, $N = 1$ ) ***	$122n$	$99n$	$93n$	$96n$
<b>Malicious setting</b>				
RS21 [27] enhanced**	$343n$	$320n$	$315n$	$318n$
Ours ( $\ell = 48$ , SH, $N = n/10$ )	$430n$	$393n$	$356n$	$332n$
Ours ( $\ell = 40$ , SH, $N = n/10$ )	$359n$	$321n$	$281n$	$253n$
Ours ( $\ell = 32$ , SH, $N = n/10$ )	$289n$	$249n$	$205n$	$175n$

\* PRTY19 has two variants, SpOT-low (lowest communication, higher computation) and SpOT-fast (higher communication, better computation). Both use expensive polynomial interpolation and require significantly more computation compared to all other protocols in this table.

\*\* Using the 3H-GCT OKVS of [13] instead of PaXoS, and the VOLE of [11] instead of the one from [32]. Setting  $\kappa_{RS21}$  to  $\kappa + \log n$  to achieve  $\kappa$  bits of security.

\*\*\* Using  $N = 1$  requires an expensive degree- $n$  polynomial interpolation.



allows to achieve malicious security without having to increase the length of the sender-to-receiver message, at the cost of increasing the receiver-to-sender communication by a factor 2. Since our approach makes this communication as low as  $O(\ell \cdot n)$ , this turns out to be an excellent tradeoff whenever the database entries are not too large. Therefore, our results show that the landscape of maliciously secure PSI is more subtle than previously thought: for large entries, the standard approach still dominates, but for smaller entries (e.g.  $\ell \leq 40$ ), the dual execution technique leads to better performances. This revives the dual execution technique, which was previously considered obsolete compared to the modern alternatives.

**Efficient PSI in the Standard Model.** Eventually, our last contribution is a new “polynomial-based” PSI protocol that does not rely on the random oracle model, following the high level structure of previous works [14, 15, 17]. To this end, we introduce the notion of PCG for the *subfield ring-OLE* correlation, and show how a simple variant of the recent PCG for ring-OLE of [7] leads to efficient instantiations of this primitive. Then, we describe a new PSI protocol built on top of this PCG, which enjoys a number of very interesting features.

*Security Features.* Our PSI protocol is in the standard model: unlike our first protocol, it does not require the random oracle model, or any tailor-made correlation-robustness assumptions. We rely solely on the (relatively well-established) ring-LPN assumption over polynomial rings with irreducible polynomials. To our knowledge, our protocol is the first standard model protocol which offers competitive performances compared to protocols using the random oracle heuristic or tailored assumptions. Furthermore, our PSI protocol enjoys full malicious security (for both parties) *almost for free*. This stems from the use of PCGs, which allows to confine the “price” of achieving malicious security to the distributed seed generation only, which has logarithmic communication and computation (in the set size  $n$ ).

We note that, though malicious security comes for free communication- and computation-wise, the tweaks used to guarantee malicious security in our protocol are not straightforward. In fact, achieving malicious security efficiently in polynomial-based PSI protocols is known to be complex and error prone. For example, previous works [14] used a superficially similar approach and claimed malicious security, but their protocol was found to be insecure in a recent preprint, which described powerful concrete attacks on this proposal [1]. Leveraging the specific structure of our protocol, we manage to get around these nontrivial subtleties with careful structural checks, for a minimal cost (independent of the database size).

*Efficiency Features.* Our PSI protocol enjoys a very low communication, considerably lower than all previous PSI protocols in the standard model which we are aware of (excluding iO- or FHE-based protocol, which can have very low communication but poor concrete efficiency). In fact, communication-wise, our PSI protocol is even on par with the best *ROM-based* PSI protocols of previous

works. Concretely, for sets of size  $n$  with  $\ell$ -bit entries, our protocol communicates  $(2\ell + 3\lambda + 3 \log n) \cdot n + o(n)$  bits. To give a single datapoint, for  $\ell = 32$  and  $n = 2^{20}$ , we estimate the total communication to be  $278n$  bits. This is on par with the best maliciously secure protocol [27], which communicates  $279n$  bits in the same setting, with comparable computation (it also uses polynomial interpolation), but without standard model security.

On Table 2, we compare our protocol to the current fastest maliciously secure PSI protocols [21, 27, 29]. As the table shows, the communication of our protocol is almost on par with that of the best protocol (the protocol of [27], enhanced with the latest VOLE protocol) for small-ish input size, and large enough set sizes. Yet, our protocol is in the standard model under the ring-LPN assumption, while [27] is only proven secure in the ROM.

**Table 2.** Comparison of the communication cost of several PSI protocols in the malicious model, for various choices of the database size  $n$  (we assume that both parties have a database of the same size) and statistical security parameter  $\lambda = 40$ , using the encoding technique of [31].  $\ell$  denote the bit-length of the inputs in the database; we set the computational security parameter  $\kappa$  to 128. For fairness of comparison, since our standard model PSI uses interpolation, we compare it to RS21 with an interpolation-based OKVS (which has better communication), and we compare our other PSIs with RS21 instantiated with (computationally) efficient OKVS.

Protocol	Communication					Hardness Assumption	Standard Model
	$n = 2^{16}$	$n = 2^{18}$	$n = 2^{20}$	$n = 2^{22}$	$n = 2^{24}$		
Our Standard PSI						Ring-LPN + OT	✓
$\ell = 64$	$724n$	$423n$	$342n$	$324n$	$323n$		
$\ell = 48$	$692n$	$391n$	$310n$	$292n$	$291n$		
$\ell = 32$	$660n$	$359n$	$278n$	$260n$	$259n$		
RS21 [27] enhanced*	$318n$	$286n$	$279n$	$279n$	$280n$	LPN + OT	✗
Our Direct PSI						LPN + OT	✗
$\ell = 64$	$421n$	$385n$	$374n$	$369n$	$365n$		
$\ell = 48$	$348n$	$311n$	$298n$	$292n$	$286n$		
$\ell = 32$	$277n$	$237n$	$223n$	$215n$	$208n$		
Our Dual PSI							
$\ell = 64$	$609n$	$535n$	$511n$	$499n$	$489n$		
$\ell = 48$	$465n$	$388n$	$361n$	$345n$	$333n$		
$\ell = 32$	$321n$	$240n$	$210n$	$192n$	$176n$		
PRTY20 [21]	$1766n$					OT	✗
RT21 [29]	$512n$					DH	✗
RS21 [27] enhanced**	$320n$	$315n$	$315n$	$317n$	$318n$	LPN + OT	✗

\* Using interpolation instead of PaXoS, and the VOLE of [11] instead of the one from [32]. Sets  $\kappa_{\text{RS21}}$  to  $\kappa + \log n$  to achieve  $\kappa$  bits of security.

\*\* Using the new OKVS of [13] instead of PaXoS, and the VOLE of [11] instead of the one from [32]. Sets  $\kappa_{\text{RS21}}$  to  $\kappa + \log n$  to achieve  $\kappa$  bits of security.

*Batch Non-interactive PSI.* On top of these security and efficiency features, the structure of our protocol allows to obtain a powerful interaction pattern: it leads to a batch non-interactive PSI, where after a short interaction with each server, a client  $C$  with set  $X$  can broadcast a *single* encoding of its database, and receive afterwards at anytime a single message from each server  $S_i$  with set  $X_i$  (plus, in the malicious setting, a small database-size-independent 2-round structural check), from which they can decode  $X \cap X_i$ . To achieve this feature, we build upon the fact that the PCG for subfield ring-OLE correlations is *programmable*, which means that we can enforce that a target party will receive the same pseudorandom string across executions with many different parties. Concretely, we achieve the following form of *batch non-interactive PSI* between a client  $C$  with database  $X$  and multiple servers  $S_i$  with datasets  $X_i$  (all of size  $n$ ):

1. In a preprocessing phase,  $C$  interacts with each of the servers, using  $O(\log n)$  communication *and* computation in each interaction, in a small constant number of rounds.
2. Then,  $C$  performs a single  $\tilde{O}(n)$  cost local computation, and broadcasts a single  $2\ell n$ -size *encoding*  $E_X$  of  $X$ .
3. Each server  $S_i$  can, at any time, send a single message  $M_i = m(X_i, E_X)$ , of length  $3(\lambda + \log n)n$ , using  $\tilde{O}(n)$  computation.
4. Eventually, given  $X$  and  $M_i$ , the client  $C$  can run a  $\tilde{O}(n)$  cost decoding procedure and recover  $X \cap X_i$ , without further interaction.

When the number of servers becomes large, our batch PSI protocol leads to strong savings for the client compared to executing a PSI protocol individually with each server. Furthermore, in this setting, the amortized communication (per PSI instance) is reduced to  $(2\ell/N_S + 3\lambda + \log n) \cdot n + o(n)$ , where  $N_S$  denotes the number of servers. Even for relatively small number of servers, the amortized communication quickly outperforms that of even the best ROM-based maliciously secure PSI protocols. For example, for  $n = 2^{24}$  and  $\ell = 32$ , the amortized communication per secure set intersection approaches  $195n$  bits with our protocol, versus  $280n$  for [27].

## 1.2 Concurrent Work

In a concurrent and independent work, recently accepted at CCS'22, Rindal and Raghuraman [25] introduced a new PSI protocol, using an approach similar to ours: the authors also leveraged subfield-VOLE to achieve communication independent of the computational security parameter  $\kappa$ . Our results have been obtained independently of theirs, around the same time period. Although their main result bears similarities to our first two contributions, we highlight some important distinctions between our work and theirs:

- The work of [25] uses an OKVS-based construction, and achieves a receiver-to-sender communication of  $(\lambda + 2 \log n) \cdot n$ . In contrast, we use a hash-based protocol, and achieve an  $(\ell - \log n) \cdot n$  receiver-to-sender communication. Therefore, we get smaller communication overall in the setting where the databases have small entries, but a slightly larger computation.

- For malicious security, the work of [25] only considers the standard paradigm of previous works (e.g. [27]), hence having a  $O(\kappa \cdot n)$  receiver-to-sender (and overall) communication. In contrast, we give two protocols, including one based on dual execution which achieves communication independent of  $\kappa$  (and smaller concrete communication for databases with small entries).
- Eventually, our last contribution, a “batchable” ring-OLE-based malicious PSI in the standard model with low communication, is unique to our work.

### 1.3 Structure of the Paper

We provide preliminaries in Sect. 2, and a detailed technical overview of our contributions in Sect. 3. Section 4 covers our ROM-based semi-honest and malicious protocols. Due to space limitation, our second malicious protocol, based on dual execution, is presented in the full version [9]. Section 5 covers our standard model PSI. Note that the additional preliminaries and all the missing proofs appear in full version [9].

## 2 Preliminaries

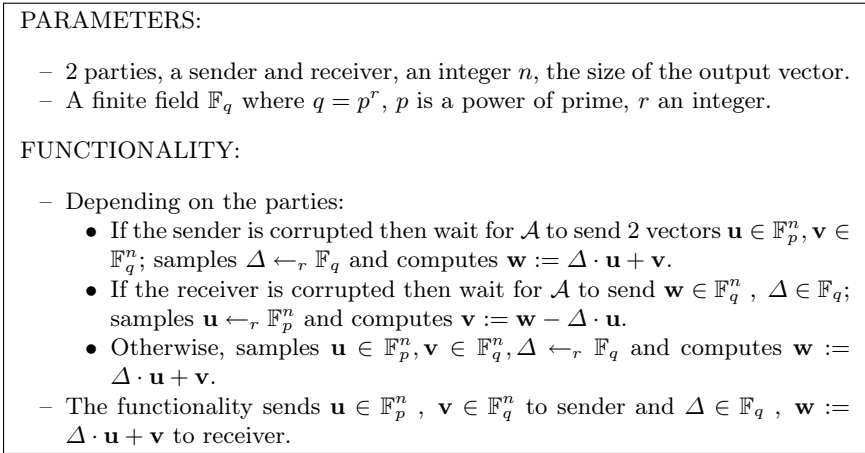
**Notation.** Throughout the paper we use the following notations: we let  $\kappa, \lambda$  denote the computational and statistical security parameters, respectively. We write  $[1, m]$  to denote a set  $\{1, 2, \dots, m\}$ . For a vector  $\mathbf{x}$  we define by  $x_i$  its  $i$ -th coordinate. Given distribution ensembles  $\{X_n\}, \{Y_n\}$ , we write  $X_n \approx Y_n$  to denote that  $X_n$  is computationally indistinguishable to  $Y_n$ .

We typically write  $\mathbb{F}_q$  to denote a field with an arbitrary subfield  $\mathbb{F}_p$ , where  $p$  is a prime power and  $q = p^t$ . We use  $\mathcal{R}_p = \mathbb{F}_p[X]/F(X)$  for the ring over the field  $\mathbb{F}_p$  where  $F(x)$  is some polynomial, and also denote  $\mathcal{R}_q = \mathbb{F}_p[X]/F(X)$ . Note that all operations in our paper are field/ring operations not modular arithmetic.

**PSI Functionality.** A private set intersection (PSI) protocol allows two parties to compute the intersection of their input sets while concealing all other information. We typically denote by  $n$  the input set sizes. For completeness, the ideal functionalities for PSI (in the semi-honest and in the malicious settings) are given in Appendix of the full version [9].

**Pseudorandom Correlation Generators (PCG).** Pseudorandom correlations generators have been introduced in a recent line of work [3–5]. A PCG allows to compress long correlations into short, correlated seeds that can later be locally expanded into pseudorandom instances of the target correlation. Slightly more formally, a PCG for a target correlation  $C$  (which samples pairs of long correlated strings  $(y_0, y_1)$ ) is a pair  $(\text{Gen}, \text{Expand})$  of algorithms such that  $\text{Gen}(1^\lambda)$  outputs a pair of short, correlated keys  $(k_0, k_1)$  and  $\text{Expand}(\sigma, k_\sigma)$  outputs a long string  $\tilde{y}_\sigma$ . Correctness states that  $(\tilde{y}_0, \tilde{y}_1)$  are indistinguishable from a random sample from  $C$ , while security states that given  $k_{1-\sigma}, \tilde{y}_\sigma$  looks like a random sample from  $C$  conditioned on satisfying the target correlation with  $\text{Expand}(1-\sigma, k_{1-\sigma})$ , for  $\sigma = 0, 1$ .

A PCG does not in itself provide a protocol to efficiently generate long pseudorandom correlations. To get the latter, one must combine a PCG with a *distributed key generation* protocol, which allows two parties to obviously run  $\text{Gen}(1^\lambda)$  such that each party gets one of the keys. Fortunately, for most PCGs of interest (and in particular, for all PCGs we use in this work), there exists very efficient low-communication distributed setup protocols [4, 7]. Combining a PCG with a distributed setup protocols allows to securely instantiate (with low communication) functionalities that distribute instances of the target correlation. In this work, we will directly rely in a black-box way on such functionalities, and use known protocols to instantiate them. We now expand on the two main functionalities we use in this work.



**Fig. 1.** Ideal functionality  $(n, p, q) - \mathcal{F}_{\text{svole}}$  of subfield vector-OLE

*Subfield Vector-OLE.* We described the subfield vector-OLE correlation in the technical overview of [9]. We represent on Fig. 1 the ideal functionality that distributes a subfield VOLE correlation. In our concrete instantiations, we will instantiate this functionality using the efficient protocol of [4]. The latter provides a general template which can be instantiated under various flavors of the LPN assumption, and provides a conservative choice under LPN for quasi-cyclic choice. A variant of LPN that leads to a considerably more efficient protocol, when plugged in the template of [4], was recently put forth in the work [11] (we note that our communications estimate are oblivious to the underlying variant: only the computational costs depends on the LPN flavor).

*Subfield Ring-OLE.* Recently, a new PCG construction was described in [7] for the *ring-OLE* correlation. The ring-OLE correlation over a ring  $\mathcal{R}_q$  is the following correlation:  $\{((x_0, z_0), (x_1, z_1)) \mid x_0, x_1, z_0 \leftarrow_r \mathcal{R}_q, z_1 \leftarrow x_0 \cdot x_1 - z_0\}$ . In this work, we rely on a slight variant of the ring-OLE correlation, where  $x_0$  is instead sampled from a subring  $\mathcal{R}_p$  of  $\mathcal{R}_q$ . We represent the corresponding variant

of the ideal functionality in the full version [9]. We note that the protocol of [7] to instantiate the ring-OLE functionality can be adapted to handle the subfield ring-OLE functionality in a straightforward way.

### 3 Technical Overview

Our starting point is the classical KKRT protocol [18], which combines Cuckoo hashing with a batch related-key oblivious pseudorandom function (BaRK-OPRF). We assume some familiarity with the KKRT protocol in this technical overview. For completeness, we provide a high level overview of KKRT, the notion of BaRK-OPRF (batch related-key oblivious pseudorandom function), and its communication costs in Appendix of full version [9]. Our construction will also rely on a functionality that distributes *subfield vector-OLE* correlation (the sVOLE functionality): Alice gets  $(\mathbf{u}, \mathbf{v})$ , and Bob gets  $(\Delta, \mathbf{w} = \Delta\mathbf{u} + \mathbf{v})$ . Such correlation can be distributed with very low communication using pseudorandom correlation generators.

#### 3.1 A New sVOLE-Based PSI for Databases with Small Entries

Subfield-VOLE leads to a simple and natural construction of BaRK-OPRF. Let  $\ell$  be the bitlength of Alice’s inputs, and let  $\mathbf{x} = (x_1, \dots, x_n)$  be the inputs of Alice, viewed as elements of  $\mathbb{F}_{2^\ell}$ . We assume for simplicity that  $\ell$  divides  $\kappa$ , the computational security parameter. Alice and Bob use an sVOLE protocol (e.g. [11]) over the field  $\mathbb{F}_{2^\kappa}$ , with subfield  $\mathbb{F}_{2^\ell}$ ; let  $(\mathbf{u}, \mathbf{v})$  be the output of Alice, and  $(\Delta, \mathbf{w})$  be the output of Bob. Recall that  $\mathbf{w} = \Delta \cdot \mathbf{u} + \mathbf{v}$ . Alice sends  $\mathbf{z} = \mathbf{x} - \mathbf{u}$  to Bob, who defines the BaRK-OPRF keys to be  $\Delta$  and  $(K_1, \dots, K_n) = \Delta \cdot \mathbf{z} + \mathbf{w}$ . The BaRK-OPRF is defined as follows:  $F_{\Delta, K_i}(y) = H(i, K_i - \Delta \cdot y)$  (all operations are over  $\mathbb{F}_{2^\kappa}$ ). Eventually, Alice outputs  $(H(i, v_i))_{i \leq n}$ . Observe that

$$\begin{aligned} H(i, v_i) &= H(i, w_i - \Delta u_i) = H(i, K_i - \Delta(z_i + u_i)) \\ &= H(i, K_i - \Delta \cdot x_i) = F_{\Delta, K_i}(x_i) \end{aligned}$$

The use of sVOLE, rather than OT extension as in the original KKRT BaRK-OPRF, has two main advantages: first, the bitwise AND is now replaced by a field multiplication. In particular, this means that we do not need anymore to use error-correcting codes, and that  $y \cdot \Delta$  retains the entire entropy of  $\Delta$ . In other words, it suffices for  $\Delta$  to be  $\kappa$ -bit long to achieve  $\kappa$  bits of security for the construction (in contrast, KKRT had to use around  $5\kappa$  bits). Second, and most importantly, the use of *subfield* VOLE allows us to completely decorrelate the size of  $\mathbf{u}$  from that of  $\Delta$ , something which can fundamentally not be achieved with the INKP OT extension. Concretely, this means that  $\mathbf{u}$  only needs to mask the input vector  $\mathbf{x}$  of Alice. If  $\mathbf{x} \in \mathbb{F}_{2^\ell}^n$ , then so do  $\mathbf{u}$  and  $\mathbf{z}$ : the communication now depends solely on the input size.

In total, our BaRK-OPRF communicates  $\ell \cdot n$  bits, plus the cost of distributing the seeds for the sVOLE generator. Using the protocol of [4] to distribute the

seeds<sup>1</sup>, the cost is logarithmic in  $n$ , hence its effect on the overall communication vanishes for large enough  $n$ .

**Combining the New OPRF with Permutation-Based Hashing.** Plugging our new BaRK-OPRF into KKRT, and using the same parameters for Cuckoo hashing, leads to a protocol with total communication  $(1.3 \cdot \ell + 3 \cdot (\lambda + 2 \log n))n + o(n)$  bits (where the  $o(n)$  terms capture the costs of distributing the PCG seeds). Concretely, for  $n = 2^{20}$  and  $\ell = 32$  (resp. 64), this already brings the cost down, from  $1008n$  bits to  $282n$  bits (resp.  $324n$  bits). However, this can be further improved using the well-established notion of *permutation-based* hashing [22]. Concretely, in *permutation-based* hashing, an item  $x$  is written as  $x_L || x_R$ , where  $x_L$  is  $\log(1.3n)$ -bit long. The item  $x$  is inserted by mapping  $x_R$  to the bin  $x_L \oplus f(x_R)$ , where  $f$  is a  $k$ -wise independent hash function, for some large enough  $k$ . This guarantees that no collision occurs, because if two items  $x, x'$  end up mapping the same value to the same bin, this means that  $x_R = x'_R$  and  $x_L \oplus f(x_R) = x'_L \oplus f(x'_R)$ , hence  $x = x'$ . When multiple hash functions are used, as in Cuckoo hashing, the index of the hash function must be appended to  $x_R$ .

Interestingly, our use of sVOLE is crucial to enabling a permutation-hashing-based optimization: the latter only provides savings when the communication involves a  $O(\ell \cdot n)$  component (which neither KKRT nor any modern OKVS-based PSI has). In our protocol, however, it further reduces the communication to  $(1.3 \cdot (\ell - \log(1.3n) + 1) + 3 \cdot (\lambda + 2 \log n))n + o(n)$  bits, which gives  $275n$  bits for  $n = 2^{20}$  and 32-bit items, or  $317n$  bits for 64-bit items. In itself, this is a really small communication improvement. However, it has an important consequence: it implies that the Alice-to-Bob communication is now completely dominated by the Bob-to-Alice communication. Concretely, this means that we can easily afford to use a much higher number of bins (which is  $1.3n$  currently) if it can allow us to reduce the number of hash functions (which is 3). This brings us to our last optimization.

**Packing Multiple Items per Bin with Generalized Cuckoo Hashing.** In this last optimization, our goal is to reduce the number of hash functions used in the Cuckoo hashing protocol, from 3 to 2, by increasing the number of bins to compensate. Unfortunately, this does not work directly with standard cuckoo hashing even while using a reasonably small stash since the cost of handling the stash is high, and nullifies all communication benefits of using two hash functions in the first place. Instead, we use a different approach: we add one degree of freedom to the Cuckoo hashing parameters, *by allowing bins to contain multiple items*. This generalization of Cuckoo hashing is not new: it has been studied in details in several works [12, 33], because it comes with a much nicer cache-friendliness than standard Cuckoo hashing.

In  $(d, k)$ -Cuckoo hashing,  $n$  items are mapped to  $(1 + \varepsilon) \cdot n$  bins using  $k$  hash functions, and each bin is allowed to contain up to  $d$  items. Allowing more items per bins significantly improves the efficiency; for example,  $(3, 2)$ -Cuckoo hashing is known to perform strictly better than standard  $(1, 3)$ -Cuckoo hashing

<sup>1</sup> This protocol uses a length- $t$  reverse VOLE protocol as a blackbox, which we instantiate with the construction of [2].

in terms of occupancy (i.e., the total number of slots  $N = d \cdot (1 + \varepsilon) \cdot n$  which must be used to guarantee a  $o(1)$  failure probability). Based on existing analysis of this variant [33], it seems reasonable to expect that (3, 2)-Cuckoo hashing already achieves a strictly smaller failure probability compared to (1, 3)-Cuckoo hashing, with a smaller number of bins.

We relied on extensive computer simulations on small values of  $n$  (from 256 to 2048) to select parameters, and extrapolated from these results parameters for larger values of  $n$ . More precisely, we ran  $10^7$  experiments with (3, 2)-Cuckoo hashing for  $n \in \{2^8, 2^9, 2^{10}\}$  (we also experimented with  $2^{11}$ , but with a smaller number of experiments) with  $c \cdot n$  bins for various values of  $c$ . Even for a value as low as  $c = 0.65$  and values of  $n$  as low as  $2^9$ , our experiments never reported any insertion failure, indicating that the empirical failure probability should already be way below  $2^{-20}$ . Since the theoretical failure probability is known to scale as  $O(1/n^\delta)$  for some constant  $\delta$  with reasonably small constant factors, we extrapolate that for large enough values of  $n$ , e.g.  $n \geq 2^{18}$ , the failure probability should be well below  $2^{-40}$ .

**Alternative Hashing Variants.** Alternatively, when allowing multiple items per bins, we can consider other hashing variants. Two natural choices are two-choice hashing [20], where each bin can have up two  $d$  items and each item is placed in the least-full of two bins, and simple hashing, where a single hash function is used to map the items to bins (standard results show that, when hashing  $n$  items to  $O(n)$  bins this way, the maximum load will be of the order of  $\log n / \log \log n$  with high probability). As we will see, these choices of hashing lead to various communication versus computation tradeoffs in our protocols, and the optimal choice also depends on the database size.

**A Membership BaRK-OPRF.** There remains a non-trivial task: to use some of the above hashing variants, we need a protocol to handle hashing with up to  $d$  items per bins. Intuitively, denoting  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(d)})$  the  $d$  entries of the bin  $i$ , we want to construct a new kind of *membership* OPRF (similar in spirit to the notion of multi-point OPRF in the literature), where Bob obtains  $F_{\Delta, K_i}(y)$  and Alice obtains the set  $F_{\Delta, K_i}(\mathbf{x}_i) = \{F_{\Delta, K_i}(x_i^{(j)})\}_{j \leq d}$ . This implies that  $F_{\Delta, K_i}(y) \in F_{\Delta, K_i}(\mathbf{x}_i)$  if and only if  $y$  is equal to any entry of  $\mathbf{x}_i$ , and  $F_{\Delta, K_i}(y)$  looks pseudorandom to Alice otherwise.

Going back to the BaRK-OPRF, recall that for a bin  $i$  where Alice placed  $x_i$  and Bob placed  $y_i$ , Alice computes  $H(i, v_i)$  and Bob computes  $H(i, K_i - \Delta y_i) = H(i, \Delta \cdot (x_i - y_i) + v_i)$ . Here, we view the  $x_i - y_i$  term as  $P_{x_i}(y_i)$ , where  $P_{x_i} = X - x_i$  is a degree-1 polynomial with root  $x_i$ . This view suggests a natural generalization of this approach, where the  $P_{x_i}$  polynomials are replaced by higher degree polynomials. Define  $P_{\mathbf{x}_i}$  to be the polynomial  $\prod_{j=1}^d (X - x_i^{(j)})$ , and let  $(c_{j,i})_{0 \leq j \leq d-1}$  denote its coefficients:  $P_{\mathbf{x}_i}(X) = X^d + \sum_{j=0}^{d-1} c_{j,i} \cdot X^j$ . Our new *membership* BaRK-OPRF is a direct generalization of the BaRK-OPRF from Sect. 3.1, which we sketch below.

**Our Construction.** Let  $m$  be the bitlength of Alice's inputs inside the bins, and let  $(\mathbf{x}_1, \dots, \mathbf{x}_N)$  be the inputs of Alice in each of the  $N$  bins, where the



inputs in each bin are viewed as length- $d$  vectors of elements of  $\mathbb{F}_{2^m}$ . We assume for simplicity that  $m$  divides  $\kappa$ , the computational security parameter. Alice and Bob use  $d$  sVOLE protocol (e.g. [11]) over the field  $\mathbb{F}_{2^\kappa}$ , with subfield  $\mathbb{F}_{2^m}$ , with the same value  $\Delta$ .<sup>2</sup> Let  $(\mathbf{u}_j, \mathbf{v}_j)_{j \leq d}$  be the outputs of Alice, and  $(\Delta, (\mathbf{w}_j)_{j \leq d})$  be the output of Bob. Recall that  $\mathbf{w}_j = \Delta \cdot \mathbf{u}_j + \mathbf{v}_j$ .

For each  $\mathbf{x}_i$ , let  $(c_{0,i}, \dots, c_{d-1,i})$  be the coefficients of the polynomial  $P_{\mathbf{x}_i}$  (omitting the coefficient of  $X^d$ , which is always 1). Let  $\mathbf{c}_j$  denote the vector  $(c_{j,i})_{i \leq N}$  for  $j = 0$  to  $d-1$ . Alice sends  $\mathbf{z}_j = \mathbf{c}_j - \mathbf{u}_j$  for  $j = 0$  to  $d-1$  to Bob, who defines the membership BaRK-OPRF keys to be  $\Delta$  and  $K_i = (k_{j,i})_{0 \leq j \leq d-1} = (\Delta \cdot z_{j,i} + w_{j,i})_{0 \leq j \leq d-1}$  for  $i = 1$  to  $N$ . Define the following degree- $d$  polynomial  $P_{\Delta, K_i}$  over  $\mathbb{F}_q$ :  $P_{\Delta, K_i}(X) = \Delta \cdot X^d + \sum_{j=0}^{d-1} k_{j,i} \cdot X^j$ . The OPRF is defined as follows:  $F_{\Delta, K_i}(y) = H(i, P_{\Delta, K_i}(y))$  (all operations are over  $\mathbb{F}_{2^\kappa}$ ). Eventually, for each bin  $i$ , Alice sets her  $d$  tuple of outputs to be  $F_{\Delta, K_i}(\mathbf{x}_i) = \{H(i, \sum_{j=0}^{d-1} v_{j,i} \cdot (x_i^{(k)})^j)\}_{k \leq d}$ . Observe that, since  $k_{j,i} = \Delta z_{j,i} + w_{j,i} = \Delta c_{j,i} + v_{j,i}$  for all  $i, j$ , we have  $H(i, P_{\Delta, K_i}(y)) = H\left(i, \Delta \cdot \left(y^d + \sum_{j=0}^{d-1} c_{j,i} y^j\right) + \sum_{j=0}^{d-1} v_{j,i} y^j\right)$ , which is equal to  $H\left(i, \Delta \cdot P_{\mathbf{x}_i}(y) + \sum_{j=0}^{d-1} v_{j,i} y^j\right)$ . Therefore, if there exists  $k \in \{1, \dots, d\}$  such that  $y = x_i^{(k)}$ , we have  $P_{\mathbf{x}_i}(y) = 0$ , and  $H(i, P_{\Delta, K_i}(y)) = H(i, \sum_{j=0}^{d-1} v_{j,i} \cdot (x_i^{(k)})^j) \in F_{\Delta, K_i}(\mathbf{x}_i)$ . On the other hand, whenever  $P_{\mathbf{x}_i}(y) \neq 0$ , then the  $\Delta \cdot P_{\mathbf{x}_i}(y)$  term in the hash makes the output pseudorandom from the viewpoint of Alice, under the correlation robustness of the hash function.

**Tying Up Loose Ends.** Using the new construction from the previous Section, together with (3, 2)-Cuckoo hashing, leads to a total communication of  $(0.65 \cdot 3(\ell - \log(0.65n) + 1) + 2 \cdot (\lambda + 2 \log n))n + o(n)$  bits, where the  $o(n)$  corresponds to the cost of setting up the PCG seeds. For  $n = 2^{20}$  and 32 bits items, this gives  $148n$  bits of communication. We mention a few remaining details. First, in the construction of membership BaRK-OPRF, Alice and Bob need to invoke  $d = 3$  length- $N$  sVOLE. In fact, it suffices to invoke a single length- $3N$  sVOLE, and to cut the output in three equal length parts, to obtain the necessary correlation. This means that the concrete cost of distributing the sVOLE seeds remains that of generating a single sVOLE (e.g.  $\approx 0.7n$  bits for  $n = 2^{20}$ ).

Second, in the above, we overlooked an important subtlety: a bin can possibly contain less than  $d$  items. In KKRT, this was handled by adding dummy items to empty bins. We use instead a more efficient approach with a negligible extra cost called a *variant* of our OPRF (details in Sect. 4).

### 3.2 Malicious Security

We then turn our attention to maliciously secure PSI. Here, it is well known that Cuckoo hashing and two-choice hashing are not usable. Consequently, we focus on simple hashing as our choice of the underlying hash technique. Using maliciously secure subfield-VOLE, which can be implemented very efficiently [4, 11],

<sup>2</sup> Note that all known sVOLE protocols allow Bob to choose the value of  $\Delta$ , hence Bob can enforce the use of the same  $\Delta$  across all instances.

we enhance our membership BaRK-OPRF to the malicious setting, with a minimal overhead. Then, we apply two standard methods to achieve security against malicious adversaries in our PSI protocol:

*First Method: Direct Approach.* The first method increases the PRF output length to  $\kappa$ . Using the analysis of [27], this suffices to allow for extracting the input of a malicious sender. However, this makes the communication depend linearly on  $\kappa$ , which severely harms communication complexity.

*Second Method: Dual Execution.* To recover a  $\kappa$ -independent communication complexity, we then turn our attention to the dual execution technique [26]. Here, the idea is simple: the parties will invoke the malicious BaRK-OPRF twice, exchanging their roles. Then, the sender sends, for each entry  $x$  of his database, a value of the form  $\text{PRF}_A(x) \oplus \text{PRF}_B(x)$ , where  $\text{PRF}_A(x)$  is obtained by the sender when invoking the BaRK-OPRF functionality as sender, and  $\text{PRF}_B(x)$  is the PRF output obtained when invoking the functionality as receiver. Here, it becomes possible to extract the input set of each party simply from its call as receiver to the BaRK-OPRF functionality, which does not require to increase the output length of the OPRF. The price to pay is that the protocol now uses two calls to the BaRK-OPRF. Concretely, the total communication becomes  $(2 \cdot N \cdot d(\ell - \log(N)) + (\lambda + \log n)n + o(n))$ , where  $N$  is the number of bins,  $d$  the maximum load of a bin, and  $\ell$  the input size (e.g. for  $n = 2^{20}$ , one can choose  $N = n/10$  and  $d = 47$ , see [26, Fig. 5]). For small database entries, this outperforms all known malicious PSI protocols.

### 3.3 An Efficient PSI in the Standard Model

In our last construction, we use a different functionality: we rely on the subfield ring-OLE functionality (given on Appendix of full version [9]), that generates a subfield ring-OLE correlation over the rings  $\mathcal{R}_p = \mathbb{F}_p[X]/F(X)$ ,  $\mathcal{R}_q = \mathbb{F}_q[X]/F(X)$ , and  $F(X)$  is some polynomial of degree  $2n + 1$  (more generally, when the two parties have sets of different size  $n$  and  $m$ ,  $F$  will be of degree  $n + m + 1$ ). At a high level, the functionality  $\mathcal{F}_{\text{sole}}$  distributes to Alice  $(a, s_A) \in \mathcal{R}_p \times \mathcal{R}_q$  and  $(b, s_B) \in (\mathcal{R}_q)^2$  to Bob such that  $ab = s_A + s_B$ . Our protocol makes a single black-box call to this functionality. Consider two parties, a sender Alice and a receiver Bob, where Alice has a set  $A = \{x_1, x_2, \dots, x_n\} \in \mathbb{F}_p^n$  and Bob has a set  $B = \{y_1, y_2, \dots, y_n\} \in \mathbb{F}_p^n$ . Define  $p_A := \prod_{i=1}^n (X - x_i) \in \mathcal{R}_p$  and  $p_B := \prod_{i=1}^n (X - y_i) \in \mathcal{R}_p$ . Let  $I := A \cap B$  denote the target output. The protocol computes the common roots of  $p_A$  and  $p_B$ , i.e.,  $\text{gcd}(p_A, p_B)$ .

By revealing appropriate linear combination of their shares and their input polynomials, Alice and Bob will “derandomize” this correlation, allowing Alice to learn the polynomial  $u = p_A b_0 + p_B b'_0$ , where  $b_0, b'_0$  are two uniformly random degree- $n$  polynomials known by Bob (this also requires revealing the high-order coefficients of  $b$ , to reduce the degree- $2n$  random polynomial  $b$  to a degree- $n$  random polynomial  $b_0$ ). Using some standard lemmas about polynomials, the polynomial  $u$  can be factored as  $\text{gcd}(p_A, p_B) \cdot p_R$ , where with high probability,  $p_R$  has no common root with  $p_A$ . This allows Alice to compute the intersection  $I = A \cap B$  as  $I = \{x_i \in A : u(x_i) = 0\}$ . Concretely:

- Alice computes and sends  $t_A = a - p_A$  to Bob.
- Bob sets  $s'_B \leftarrow s_B - t_A b$ . Then, Bob decomposes  $b$  as  $b = b_0 + b_1 \cdot X^n$  (where  $b_0, b_1$  are degree- $n$  polynomials), sets  $s'_B \leftarrow s_B - t_A b$ , and picks a random degree- $n$  polynomial  $b'_0$  over  $\mathcal{R}_q$ . He sends  $b_1$  and  $t_B \leftarrow s'_B + p_B b'_0$  to Alice.
- **(Output)** Alice sets  $u \leftarrow t_B - p_A b_1 \cdot X^n + s_A$ ; note that  $u = p_A b_0 + p_B b'_0$ . Alice outputs the set  $I = \{x \in A \mid u(x) = 0\}$ .

We prove that this construction achieves “augmented semi-honest security”, a strengthening of honest-but-curious corruption where the adversary is allowed to change the corrupted parties’ inputs. Furthermore, we securely realize the functionality  $\mathcal{F}_{\text{sole}}$  using the PCG-based protocol of [7], which is secure under the ring-LPN assumption. Instantiating the subfield ring OLE this way allows to import a powerful feature of the PCG of [7], which is its *programmability*: when generating a ring-OLE correlation, the receiver can ensure that her output  $a$  remains *identical* across multiple instances of the protocol with different parties. Using this programmability feature, we show that our protocol can be *batched*: a single  $O(\ell \cdot n)$ -size client message encoding her database  $A$  can be reused with  $N$  different servers with databases  $B_i$ , allowing her to learn  $A \cap B_i$  using a single message from each server afterwards.

**Achieving Malicious Security.** We then turn our attention to security against malicious adversaries. Our upgrade introduces only a minimal communication overhead to the protocol, independent of the set sizes  $n$ . At a high level, the main issues that can occur in the malicious setting is when Alice sets  $p_A = 0$ , or when Bob sets  $p_B b'_0 = 0$ . Indeed, since Alice gets  $u = p_A b_0 + p_B b'_0$ , if  $p_A = 0$ , she can learn Bob’s entire input set  $p_B$ . On the other hand, if  $p_B b'_0 = 0$ , Bob forces the output to be  $A$ .

We handle both issues separately. The second issue is intuitively simpler to handle, since when Bob carries out this attack, Alice will notice that her output is exactly her set  $A$ . This suggests a simple way around: if Alice notice at the end of the protocol that the output is equal to  $A$ , she aborts the protocol. Of course, a honest Bob could have an input  $B$  with  $A \subseteq B$ , in which case this modification would harm correctness. But there is a simple way around: prior to the protocol, Alice and Bob can just agree on a reserved dummy item  $d$  (we will pick  $d = 1$  in the protocol, but this choice is arbitrary), which is guaranteed to be in neither databases. If database entries are elements of a field  $\mathbb{F}_{p'}$ , this can simply be done by choosing any slightly larger field  $\mathbb{F}_p$  of size  $|\mathbb{F}_p| \geq |\mathbb{F}_{p'}| + 1$ , reserving one element of  $\mathbb{F}_p$  to encode  $d$ , and mapping the elements of  $\mathbb{F}_{p'}$  to the remaining elements. Then, Alice and Bob execute the protocol on inputs  $A \cup \{1\}$  and  $B$ , which guarantees that  $B$  does not contain  $A$ .

For the first issue, Bob must check before sending  $t_B = s'_B + p_B b'_0$  that Alice did not set  $p_A$  to be 0 when computing  $t_A = a - p_A$ . Intuitively, this will be done by letting Bob check that  $p_A(x) \neq 0$ , for an appropriate input  $x$ . This, however, must be done with some care, since learning  $p_A(x)$  could leak information to a corrupted Bob. We handle this issue by reserving a second element of  $\mathbb{F}_p$  (hence we now need  $|\mathbb{F}_p| \geq |\mathbb{F}_{p'}| + 2$ ), which we assume w.l.o.g. to be 0, which should again be in neither set. Then, Alice will define the encoding of her set to be

the degree- $n$  polynomial  $p_A$  such that  $p_A(\text{map}(a)) = 0$  for every  $a \in A$ , and  $p_A(0) = 1$ . Then, we let Bob first send  $b_1$ , without sending  $t_B$ . Afterwards, Bob computes  $s'_B \leftarrow s_B - t_A b$  and Alice computes  $s'_A \leftarrow s_A - p_A b_1 \cdot X^n$ . Observe that if both parties behave honestly,  $s'_a + s'_b = ab - t_A b - p_A b_1 \cdot X^n = ab - ab + p_A b - p_A b_1 \cdot X^n = p_A b_0$ . To enforce  $p_A \neq 0$ , we will check that the above equation holds for some nonzero  $p_A$ . Crucially, since both  $p_A$  and  $b_0$  have degree at most  $n$ , no reduction modulo  $F(X)$  occurs in the right hand side of the equation. This implies that we can simply check that the equation holds for the reserved input  $x = 0$  (since a honest  $p_A$  is guaranteed to satisfy  $p_A(0) = 1 \neq 0$ ). To check this, we let Alice send  $s'_A(0)$  to Bob, who checks that  $s'_A(0) = b_0(0) - s'_B(0)$ ; if the check fails, Bob aborts the protocol.

## 4 PSI from Subfield-VOLE

### 4.1 A New Membership Batched OPRF

Our BaRK-OPRF allows the sender to hold a set of keys  $(\mathbf{k}_i)_{i \leq N}$  such that each key is assigned with a tuple of  $d$  input elements of the receiver and then the receiver learns a PRF output on each element in this tuple corresponding with the same key. More formally, denoting  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(d)})$  consisting of  $d$  entries, the sender gets  $F(i, y)$  and the receiver obtains a set  $\{F(i, x_i^{(j)})\}_{j \leq d}$  such that  $F(i, y) \in \{F(i, x_i^{(j)})\}_{j \leq d}$  if and only if  $y$  is equal to any entry of  $\mathbf{x}_i$ , and  $F(i, y)$  looks pseudorandom to the receiver otherwise.

**PARAMETERS:**

$\mathbb{F}_p$  is a finite field. There are 2 parties, a sender and a receiver with input set  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subseteq \mathbb{F}_p$  where  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(d)})$ .

**FUNCTIONALITY:**

- Wait for input (**sender, id**) from the sender and (**receiver, id, X**) from the receiver. The functionality samples a PRF  $F$  then  $\forall x \in \mathbf{x}_i$  outputs  $F(i, x)$  to the receiver for  $i \in [1, N]$ .
- When the sender inputs any  $(i, y) \in [1, N] \times \mathbb{F}_p$ , functionality gives  $F(i, y)$  to the sender.

**Fig. 2.** Ideal functionality  $\mathcal{F}_{\text{oprf}}$

**Main Construction.** Assume that the receiver inputs the set of  $n = Nd$  elements:  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subseteq \mathbb{F}_p$  where  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(d)})$ . First, the sender and the receiver invoke the  $\mathcal{F}_{\text{svole}}$  protocol of dimension  $n$ , with their roles reversed, to get a random sVOLE correlation. Specifically, the receiver learns a pair of vectors  $(\mathbf{u}, \mathbf{v})$  where  $\mathbf{u} \in \mathbb{F}_p^n$ ,  $\mathbf{v} \in \mathbb{F}_q^n$ , the sender gets  $\Delta \in \mathbb{F}_q$  and  $\mathbf{w} := \Delta \cdot \mathbf{u} + \mathbf{v}$ . Denoting  $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N)$  where  $(u_{j,i})_{1 \leq j \leq d}$  are  $d$  entries of vector  $\mathbf{u}_i$ . This notation is the same for  $\mathbf{v}, \mathbf{w}$ . Consider  $\mathbf{x}_i$  and its associated polynomial as  $P_{\mathbf{x}_i}(X) = \prod_{j=1}^d (X - x_i^{(j)}) = X^d + \sum_{j=1}^d c_{j,i} \cdot X^{j-1}$  where  $c_{j,i} \in \mathbb{F}_p$  for  $i \in [1, N], j \in [1, d]$ .

Now, the receiver defines  $\mathbf{c}_i := (c_{j,i})_{j \leq d}$ ,  $\mathbf{c} := (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N)$ , and then  $\forall i \in [1, N]$  sends to the sender  $\mathbf{z}_i := \mathbf{c}_i - \mathbf{u}_i \in \mathbb{F}_p^d$ . Above, the  $\mathbf{u}_i$  are masks for the coefficients  $\mathbf{c}_i$  of (the polynomial associated)  $\mathbf{x}_i$ . Indeed,  $\mathbf{u}_i$  are distributed uniformly at random in the subfield  $\mathbb{F}_p$ , then the vector  $\mathbf{z}_i$  is a uniformly random over  $\mathbb{F}_p^d$  from the viewpoint of the sender. The two parties will run a coin flipping protocol to get a random value  $t \leftarrow \mathbb{F}_q$ . For  $i \in [1, N]$ , the receiver defines the PRF output on each input  $x \in \mathbf{x}_i$  as  $F(i, x) = H\left(i|t|x, \sum_{j=1}^d v_{j,i} \cdot x^{j-1}\right)$ .

On the other hand, after receiving the vectors  $\mathbf{z}_i$ , for  $i \in [1, N]$ , the sender defines the vector  $\mathbf{k}_i := \mathbf{w}_i + \Delta \cdot \mathbf{z}_i$ . As a consequence, for any input  $(i, y) \in [1, N] \times \mathbb{F}_p$ , its PRF output is computed as:  $F(i, y) = H\left(i|t|y, \Delta \cdot y^d + \sum_{j=1}^d k_{j,i} \cdot y^{j-1}\right)$ .

**Correctness and Security.** To see why PRF output is defined as above. Observe that  $\mathbf{k}_i := \mathbf{w}_i + \Delta \cdot \mathbf{z}_i = \mathbf{v}_i + \Delta \cdot \mathbf{c}_i$ . Then, we have

$$\begin{aligned} \Delta \cdot y^d + \sum_{j=1}^d k_{j,i} \cdot y^{j-1} &= \Delta \cdot y^d + \sum_{j=1}^d (v_{j,i} + \Delta \cdot c_{j,i}) \cdot y^{j-1} \\ &= \Delta \cdot (y^d + \sum_{j=1}^d c_{j,i} \cdot y^{j-1}) + \sum_{j=1}^d v_{j,i} \cdot y^{j-1} = \Delta \cdot P_{\mathbf{x}_i}(y) + \sum_{j=1}^d v_{j,i} \cdot y^{j-1} \end{aligned}$$

so if  $y \in \mathbf{x}_i$  then  $P_{\mathbf{x}_i}(y) = 0$  which leads to  $F(i, y) \in \{F(i, x_i^{(j)})\}_{j \leq d}$ .

**Theorem 1.** *The protocol  $\Pi_{\text{opr}} (Fig. 3)$  instantiated with random oracles  $H, H'$ , securely realizes the ideal functionality of  $\mathcal{F}_{\text{opr}} (Fig. 2)$  against a malicious setting in the  $\mathcal{F}_{\text{vole}}$  hybrid model.*

Note that the output  $v$  of  $H$  is chosen depending on the concrete structure of PSI and the target setting (semi-honest or malicious). This parameter is detailed in the Sect. 4.2 for a semi-honest setting and the Sect. 4.3 for a malicious setting.

## 4.2 A New Semi-honest PSI from mOPRF

**A Variant of BaRK-OPRF.** We now propose a variant of our BaRK-OPRF to deal with the case when the size of each tuple input is not necessarily equal to  $d$ . This means that the receiver now can divide the input set to  $N$  tuples  $\mathbf{x}_i$  and each tuple has less than or equal to  $d$  items. Meanwhile, the sender is not allowed to learn about how many exactly items are in each tuple. This functionality can be obtained from our BaRK-OPRF plus a small extra cost, i.e., a *subfield* VOLE of length  $N$  over the subfield  $\mathbb{F}_2$ .

The idea is as follows. The receiver’s input set  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subseteq \mathbb{F}_p$  where  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(j_i)})$ ,  $j_i \leq d$ . The polynomial associated to  $\{\mathbf{x}_i\}_{i \leq N}$  will be expressed as a polynomial of degree  $d$ :  $P_{\mathbf{x}_i}(X) = \prod_{j=1}^{j_i} (X - x_i^{(j)}) = \sum_{j=1}^{d+1} c_{j,i} \cdot X^{j-1}$  where  $c_{j,i} \in \mathbb{F}_p$ .

As a result, the set of the coefficients of  $P_{\mathbf{x}_i}(X) = (c_{1,i}, c_{2,i}, \dots, c_{d+1,i})$ . We remark that, compared to the associated polynomial in our original BaRK-OPRF which has a constant coefficient of degree  $d$  of 1, in our variant version

**PARAMETERS:**

- Given  $\mathbb{F}_p \subseteq \mathbb{F}_q$  where  $\mathbb{F}_q \approx O(2^\kappa)$ ,  $\mathbf{H} : \{0, 1\}^* \times \mathbb{F}_q \rightarrow \{0, 1\}^v$  and  $\mathbf{H}' : \mathbb{F}_q \rightarrow \mathbb{F}_q$  are random oracles.
- The sender has no input and the receiver inputs a set  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subseteq \mathbb{F}_p$  where  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(d)})$  and  $n = Nd$ .

**PROTOCOL:**

1. The sender and the receiver invoke to the  $\mathcal{F}_{\text{svole}}$  of dimension  $n$  in the  $\mathbb{F}_q$  over the  $\mathbb{F}_p$  with the inverse role. The receiver gets two random vectors  $\mathbf{u} \in \mathbb{F}_p^n, \mathbf{v} \in \mathbb{F}_q^n$  and the sender receives  $\Delta \in \mathbb{F}_q, \mathbf{w} := \Delta \mathbf{u} + \mathbf{v} \in \mathbb{F}_q^n$ . Denoting  $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N)$  where  $\mathbf{u}_i = (c_{j,i})_{1 \leq j \leq d}$ . This denotation is the same for  $\mathbf{v}, \mathbf{w}$ .
2. The receiver samples  $t_r \leftarrow \mathbb{F}_q$  and sends  $h_r := \mathbf{H}'(t_r)$  to the sender.
3. The sender samples  $t_s \leftarrow \mathbb{F}_q$  and sends  $h_s := \mathbf{H}'(t_s)$  to the receiver.
4. The receiver determines the associated polynomial for each  $\mathbf{x}_i$  as

$$P_{\mathbf{x}_i}(X) = \prod_{j=1}^d (X - x_i^{(j)}) = X^d + \sum_{j=1}^d c_{j,i} \cdot X^{j-1}$$

where  $c_{j,i} \in \mathbb{F}_p$  for  $i \in [1, N], j \in [1, d]$ .

5. Denoting  $\mathbf{c}_i := (c_{j,i})_{1 \leq j \leq d}; \mathbf{c} := (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N)$ , the receiver computes  $\mathbf{z}_i := \mathbf{c}_i - \mathbf{u}_i \in \mathbb{F}_p^d$ , and then sends  $\mathbf{z}_i$  and  $t_r$  to the sender.
6. The sender aborts if  $\mathbf{H}'(t_r) \neq h_r$ .
7. The sender sends  $t_s$  to the receiver, the receiver aborts if  $\mathbf{H}'(t_s) \neq h_s$  and both parties define  $t = t_s \oplus t_r$ .
8. The receiver outputs the PRF values on the input  $x \in \mathbf{x}_i$  for  $i \in [1, N]$  as

$$F(i, x) = \mathbf{H} \left( i|t|x, \sum_{j=1}^d v_{j,i} \cdot x^{j-1} \right)$$

9. For  $i \in [1, N]$ , the sender defines  $\mathbf{k}_i = \mathbf{w}_i + \Delta \mathbf{z}_i$ . For any input  $(i, y) \in [1, N] \times \mathbb{F}_p$ , the sender computes the PRF output by below formula

$$F(i, y) = \mathbf{H} \left( i|t|y, \Delta \cdot y^d + \sum_{j=1}^d k_{j,i} \cdot y^{j-1} \right)$$

**Fig. 3.** Our batch BaRK-OPRF  $\Pi_{\text{oprf}}$  based on subVOLE

this coefficient will equal 0 or 1 since the degree of  $P_{\mathbf{x}_i}(X)$  is *less* than or equal to  $d$ . So, it requires  $(d + 1)$  masks for this polynomial instead of  $d$ , but the mask for the coefficient of degree  $d$  only needs to be in  $\mathbb{F}_2$ . For each tuple, we require an additional value  $u_i \in \mathbb{F}_2$ , so in total we need an additional subfield VOLE of length  $N$  over the subfield  $\mathbb{F}_2$ .

More formally, the sender and receiver invoke a subfield VOLE of length  $n$  over the subfield  $\mathbb{F}_p$  as before (all the notations in Fig. 3 are reused), and additionally invoke another subfield VOLE instance over the subfield  $\mathbb{F}_2$  of length  $N$  with

an inverse role, while the receiver gets  $\mathbf{u}' \in \mathbb{F}_2^N$ , and  $\mathbf{v}' \in \mathbb{F}_q^N$  the sender holds  $\Delta \in \mathbb{F}_q$  ( $\Delta$  is the same for each time invoking *subfield* VOLE) and  $\mathbf{w}' := \Delta \cdot \mathbf{u}' + \mathbf{v}'$ . The receiver sends to the sender vectors  $\mathbf{z}_i$  as before, and an extra vector  $\mathbf{z}'$  defined as  $z'_i := c_{d+1,i} - u'_i$  for  $i \in [1, N]$ . The receiver outputs on input  $x \in \mathbf{x}_i$  are computed as  $F(i, x) = H(i|t|x, v'_i \cdot x^d + \sum_{j=1}^d v_{j,i} \cdot x^{j-1})$ . On the other hand, the sender defines their PRF values on input  $(i, y)$  where  $i \in [1, N]$ ,  $y \in \mathbb{F}_p$  as  $F(i, y) = H(i|t|y, (w'_i + \Delta z'_i) \cdot y^d + \sum_{j=1}^d k_{j,i} \cdot y^{j-1})$ .

**Main Construction of a New PSI.** The sender and the receiver have two input sets  $X = \{x_1, x_2, \dots, x_n\}$  and  $Y = \{y_1, y_2, \dots, y_n\}$ . Assume that all of these elements have the bit-length  $\ell$ . Intuitively, our BaRK-OPRF is constructed from *sub*VOLE to handle the case when having multiple items per bin. Then this specialized BaRK-OPRF can combine with some hashing techniques to form an efficient PSI protocol. In the next part 4.2, we discuss these types of hashing. Our PSI protocol is described in Fig. 4; it builds upon the protocol of [18] using GCH and BaRK-OPRF. For simplicity, we describe our protocol directly with generalized Cuckoo hashing; adapting the protocol to other variants is immediate. We elaborate on our protocol below. In our protocol, the receiver first uses  $(d, k)$ -Cuckoo hashing to map his input set  $Y$  to a table with  $N$  bins, note that the bit-length of the values stored in a bin is  $\ell - \log N$  instead of  $\ell$ . Depending on the size of  $n$ , we use one of two approaches to handle the bins which are not full (the threshold was chosen empirically to optimize communication).

- If  $n \geq 2^{20}$ , the variant of our BaRK-OPRF (using an additional subfield VOLE over  $\mathbb{F}_2$ ) is used; for such sizes, the concrete cost of implementing the additional *s*VOLE vanishes.
- Otherwise, when  $n < 2^{20}$ , the receiver adds dummy items to bins such that each bin contains *exactly*  $d$  items. To avoid collisions between the dummy items and the elements in the same bin of the sender, we pad an extra bit to all items in the following way:  $i|x|b$  where  $i$  is the index of hash function corresponding with the stored value  $x$  while  $b = 1$  if  $x$  is a dummy item added and  $b = 0$  otherwise.

In both case, the sender computes  $k \cdot n$  PRF evaluations and sends (shuffled) to the receiver, who compares them with his OPRF outputs, and outputs the intersection set. To reduce the computational cost in this step, the sender can send separately each set  $H_i$  ( $i \in [1, k]$ ) which contains the PRF outputs of each  $x \in X$  with the related bin  $h_i(x)$ . Then for each element, the receiver only needs to search for one set (among  $k$  sets  $H_i$ ) of  $n$  items instead of  $k \cdot n$ .

**Alternative Hashing Methods.** There are two hashing schemes that can be fit into our PSI structure.

*2-choice hashing* [20] is a variant of Cuckoo hashing where one item  $x$  is assigned to one of two bins  $h_1(x)$  or  $h_2(x)$ . However, there is no restriction on the number of items per bin and an item is put in a bin which already has fewer items. [20] proposes both theoretical references and heuristic parameters

## PARAMETERS:

- The sender and the receiver have respectively input sets  $X = \{x_1, x_2, \dots, x_n\}$  and  $Y = \{y_1, y_2, \dots, y_n\}$ , all elements of bit-length  $\ell$ .
- A  $(d, k)$ -generalized Cuckoo hashing (GCH) scheme mapping  $n$  items to  $N$  bins by  $k$  hash functions  $h_1, h_2, \dots, h_k : \{0, 1\}^* \rightarrow [N]$  where  $Nd > n$  and  $d = O(1)$  (see Sect. 4.2).

## PROTOCOL:

1. The receiver uses  $(d, k)$ -Cuckoo hashing with  $k$  hash functions to map the elements in  $Y$  to the table  $\mathcal{B}$  consisting of  $N$  bins, where each bin  $i$  has  $j_i \leq d$  items. Denote  $y_{j,i}$  is an element in  $Y$  assigned to position  $j$  of bin  $i$  and its stored value in table  $\mathcal{B}$  is  $y'_{j,i}$ .
2. Depending on the size of  $n$ , there are two alternatives:
  - (a)  $n \geq 2^{20}$ , the sender and receiver invoke our variant of  $\Pi_{\text{oprf}}$  where the receiver uses the input set  $Y_{\mathcal{B}} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$  defined as follows:
    - $\mathbf{y}_i = \{r_{1,i}, r_{2,i}, \dots, r_{j_i,i}\}$ .
    - $r_{j,i} = t \parallel y'_{j,i}$  where  $t$  is index of a hash function such that  $h_t(y_{j,i}) = i$ .
  - (b)  $n < 2^{20}$ , the sender and receiver directly invoke the  $\Pi_{\text{oprf}}$  where the receiver uses the input set  $Y_{\mathcal{B}} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$  defined as follows:
    - $\mathbf{y}_i = \{r_{1,i}, r_{2,i}, \dots, r_{d,i}\}$ .
      - For  $j \leq j_i$ :  $r_{j,i} = t \parallel y'_{j,i} \parallel 1$  where  $t$  is index of hash function such that  $h_t(y_{j,i}) = i$ .
      - Otherwise,  $r_{j,i} = t \parallel \text{dummy value} \parallel 0$  where  $t \leftarrow_r [1, k]$ .
3. The receiver obtains  $n$  instances OPRF:

$$Y' = \{\text{PRF}(i, r_{i,j}) \mid i \in [1, N], j \leq j_i\}$$

4. The sender uses the  $k$  hash functions to map the  $n$  element in  $X$  to the  $N$  bins. Let  $x_t$  denote the value stored at bin  $h_t(x)$  when mapping  $x$  for  $t \in [1, k]$ .
5. The sender computes the sets of  $k \cdot n$  PRF outputs:
  - (a) For  $n \geq 2^{20}$ :  $H_t = \{\text{PRF}(h_t(x), t \parallel x_t) \mid x \in X\}$  for  $t \in [1, k]$ .
  - (b) For  $n < 2^{20}$ :  $H_t = \{\text{PRF}(h_t(x), t \parallel x_t \parallel 1) \mid x \in X\}$  for  $t \in [1, k]$ .
 Then the sender randomly permutes and sends each set to the receiver.
6. The receiver finds the intersection:
  - if  $y \in Y$  is mapped to the position  $j$  of bin  $i$  by function  $h_t$  then check whether  $\text{PRF}(i, r_{i,j}) \in H_t$  ( $r_{i,j}$  is defined depending on  $n$ ).
  - Outputs the intersection set.

**Fig. 4.** Our new semi-honest PSI protocol from BaRK-OPRF

for 2-choice hashing, which require only a small number of dummy items. Let us assume we have  $n$  items and 2 hash functions; using 2-choice hashing allows to map  $n$  items to  $N$  bins in time  $O(n \log n)$  where each bin contains at most  $L = \lceil n/N \rceil + 1$  items with a probability  $1 - O(1/N)^{L-1}$ .

*Simple hashing* uses one hash function  $h$  to map an item  $x$  to bin  $h(x)$ . For security, the number of items per bin can leak some information then it requires



padding each bin with dummy items until having an equal number of items per bin. With very high probability, for  $N = O(n \log n)$  bins, the maximum possible items per bin is  $O(\log n)$ . The percentage of the occupation of dummy items is higher than others. However, simple hashing avoids ambiguities about where an item can be placed, a property which is crucial in the malicious setting.

**Parameters.** In this section, we discuss concrete parameters used in our new PSI semi-honest protocol. We use  $\kappa = 128$  and  $\lambda = 40$ . The protocol contains several parameters:

*The Length of OPRF Output.* The output domain of PRF would be  $\{0, 1\}^v$  where  $v = \lambda + 2 \log_2(n)$  guarantees a  $2^{-\lambda}$  bound on the collision probability of PRF outputs among the two size- $n$  sets. Furthermore, communicating the hashes can be reduced to communicating only  $\approx \lambda + \log n$  bits per hash, using a heuristic technique of [31] that directly leads to an optimization of our PSI protocol.

*The size of  $\mathbb{F}_p$  and  $\mathbb{F}_q$  in BaRK-OPRF.* After using permutation-based hashing, each element is mapped to a bin with a stored value in this bin, the bit-length reduces from  $\ell$  to  $\ell - \log N$ . The input set of BaRK-OPRF in PSI protocol constructs from stored values concatenating with some extra bits. Then the bit-length of an input element of BaRK-OPRF is computed as  $\ell - \log N + 1$  if  $n \geq 2^{20}$  or  $\ell - \log N + 2$  otherwise, i.e., the size of  $q = 2^{\ell - \log N + 1}$  or  $q = 2^{\ell - \log N + 2}$  respectively.

*Generalized Cuckoo Hashing.* We use a  $(d, k)$ -general cuckoo hashing scheme without stash. The parameters are chosen such that the failure probability is  $2^{-\lambda}$ . When  $d = 1, k = 3$  these parameters are identical with KKRT except for the number of bins increases slightly to  $N = 1.3n$  which is a trade-off to obtain no stash. Even with the higher number of bins, our PSI protocol significantly outperforms KKRT.

To minimize the overall communication, we set  $k = 2$  to reduce the cost of sending  $k \cdot n$  PRF outputs. We used a Python script to simulate randomly assigning  $n$  values to  $N = c \cdot n$  bins using  $(d, 2)$ -Cuckoo hashing, for several values of  $d$  and  $c$ , and for  $n = 2^9, 2^{10}, 2^{11}, 2^{12}$ . For a value of  $c$  as low as 0.65, we never observed any insertion failure over  $10^7$  trials for each values of  $n$  (for  $n = 2^{12}$ , we could only do  $10^6$  trials), when using  $d = 3$  items per bins. For  $d = 2$ , the failure probability became noticeable already for  $c \approx 1$ . Based on known theoretical analysis of  $(d, k)$ -Cuckoo hashing, the failure probability is known to scale inverse polynomially with  $n$ . Therefore, we expect that for reasonably large values of  $n$  (e.g.  $n \geq 2^{18}$ ), our parameters should guarantee a failure probability significantly below  $2^{-40}$ .

*2-Choice Hashing.* Following the analysis of [20], we set the number  $N$  of bins to  $n/3$ , and the maximum load  $d = L + 1$  to 4. This guarantees a failure probability which we empirically estimate to be  $1/N^{L-1}$ , which is below  $2^{-40}$  for all values of  $n$  above  $2^{14}$ .

*Simple Hashing.* Eventually, for simple hashing, we set arbitrarily the number of bins  $N$  to  $n/10$ , and derive the corresponding value of  $d$  from Fig. 5 in [26]. We note that the parameters for simple hashing are much less heuristic than the other two, in that concrete bound can actually be achieved which are relatively close to the heuristic (computer-estimated) bounds. For example, [20] experimentally observes that for a  $2^{-40}$  failure probability, setting  $d = 47$  suffices when using  $N = n/10$  bins. Using a standard Chernoff bound, it is in fact straightforward to prove formally that  $d = 49$  already suffices to reach this failure probability, which is very close to the experimental bound. In contrast, experimental bounds in more complex hashing variants are typically much more distant from provable bounds. The choice of  $N = n/10$  is entirely arbitrary: any smaller  $N$  leads to better communication, but requires using higher values of  $d$ , leading to worse computation (due to the need to perform  $N$  polynomial interpolations with degree- $d$  polynomial). This allows for a smooth tradeoff between communication and computation, where better computational power can be used to further reduce the communication. At the extreme end of the spectrum, using  $N = 1$  and  $d = n$  requires one expensive degree- $n$  polynomial interpolation, but can achieve extremely low communications, e.g.  $93n$  bits of communication for  $\ell = 32$  and  $n = 2^{20}$ .

**Efficiency.** We compare the communication of our protocols, using three hashing methods, on Table 1. Regarding computation, we provide a breakdown of the computation costs of our protocols in the Appendix of full version [9]. Briefly, though, compared to the protocol of [27], and when using a standard choice of parameters for our protocol (e.g.  $n = 2^{20}$ , and using generalized Cuckoo hashing with  $d = 3$  and  $N = 0.65n$ ), our protocol requires essentially a length- $1.9n$  VOLE (with a small subfield),  $0.65n$  degree-3 polynomial interpolations (roughly  $3n$  multiplications over a small field), and computing  $n$  hashes. In contrast, the enhanced version of [27] (using the OKVS of [13] and the VOLE of [11]) will require solving a linear system to set up an OKVS (this requires on the order of  $(1.3 \log n + \lambda)^3$  multiplications over  $\mathbb{F}_{2^{128}}$ , plus  $O(\lambda n)$  operations), computing a length- $1.3n$  VOLE (over  $\mathbb{F}_{2^{128}}$ ), and computing  $2n$  hashes. The cost of the VOLE dominates that of performing  $n$  hashes, so for sufficiently large set sizes ( $n \gg 2^{20}$ ), the protocol of [27] should become roughly 30% more efficient than our protocol computation-wise. For smaller sets (e.g.  $n \approx 2^{16}$ ), the cost of setting up the OKVS becomes more significant, requiring around  $20n$  field multiplications over  $\mathbb{F}_{2^{128}}$ , hence the computational efficiency of our protocol becomes roughly on par with that of [27]. Of course, real runtimes can vary due to e.g. cache misses, so these estimations should only be viewed as a first order approximation indicating that the computational efficiency of our protocols is close to that of [27] (but likely slightly larger).

In terms of computation, the main computational overhead comes from performing  $N$  polynomial interpolations of *only degree- $d$*  polynomials. Based on our analysis, to achieve  $2^{-\lambda} = 2^{-40}$  probability of insertion failure, the following parameters can be chosen:

- $N = 0.65n$  and  $d = 3$  for generalized Cuckoo hashing (GCH),
- $N = 0.33n$  and  $d = 4$  for two-choice hashing,
- $N = n/10$  and  $d \approx 46$  for simple hashing.

As the above illustrates, the cost of performing  $N$  polynomial interpolations will be very small for GCH, two-choice hashing, but becomes higher for simple hashing (though performing  $n/10$  degree-46 interpolations remains reasonably fast).

### 4.3 A Malicious PSI from mOPRF

In this section, we propose a maliciously secure PSI protocol based on our BaRK-OPRF (Sect. 4.1) and simple hashing combining a permutation-based hash function. The PSI protocol is shown in Fig. 5 and its security against a corrupted adversary is proven in Theorem 2. The estimated overhead communication cost of this PSI is  $Nd(\ell - \log N) + (\kappa + \log n)n + o(n)$ . Observe that the PSI protocol in Sect. 4.2 is insecure against malicious settings since the general hashing scheme does not allow the simulation in ideal world. To handle this we use simple hashing schemes with only one permutation-based hash function. This protocol is constructed from the natural approach used recently in [10, 20, 21, 27], i.e., Alice (a sender) and Bob (a receiver) invoke the  $\mathcal{F}_{\text{oprf}}$  then Bob gets the PRF values on his input and Alice enables to compute the PRF on any input so Alice computes on her input after that she sends these PRF values to Bob; Bob compares and outputs the intersection.

**PARAMETERS:**

- Alice (sender) and Bob (receiver) have respectively input set  $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{F}_p$  and  $Y = \{y_1, y_2, \dots, y_n\} \in \mathbb{F}_p$ , all elements of bit-length  $\ell$ .
- A random hash functions  $h : \{0, 1\}^* \rightarrow [N]$ .
- A Permutation-based hashing  $\text{Per}_{h,X}$  maps a set  $X$  to table  $\mathcal{B}_X$  consisting of  $N$  bins such that each bin has  $d$  slots where  $Nd > |X|$ , and  $d = O(1)$ . Denote  $\text{Per}(x) := (i, x')$  where  $x'$  is the stored value of  $x$  in bin  $i$  which defined by  $h$  and  $x$  then  $\text{Per}^{-1}(i, x') = x$ .

**PROTOCOL:**

1. Bob uses  $\text{Per}$  to map  $Y$  to  $\mathcal{B}_Y$ , for each empty slot in each bin  $\mathcal{B}_Y[i]$ , put here a dummy item of length  $\ell - \log N$ .
2. Alice sends (sender, id) and Bob sends (receiver, id,  $\mathcal{B}_Y$ ) to  $\mathcal{F}_{\text{oprf}}$  then
  - Bob receives the  $Y' = \{F(i, y') \mid y' \in \mathcal{B}_Y[i]\}_{i \leq N}$ .
3. For each  $x \in X$ , Alice queries  $x$  to  $\mathcal{F}_{\text{oprf}}$  with corresponding input  $(i, x')$  such that  $\text{Per}(x) = (i, x')$ , then Alice gets  $F(i, x')$ . Alice sends to Bob

$$U = \{F(i, x') \mid x \in X \wedge \text{Per}(x) = (i, x')\}$$

4. Now for each  $y \in Y$ ,  $\text{Per}(y) = (i, y')$ , if  $F(i, y') \in U$  then Bob outputs  $y$  as an element in the intersection.

**Fig. 5.** Our malicious PSI protocol based on  $\mathcal{F}_{\text{oprf}}$

Intuitively, in a malicious setting, when the sender is corrupted, the simulation needs to extract the sender's input set  $X$  from the queries to  $\mathcal{F}_{\text{opr}}f$  and the set  $U$ . Denote  $F(y) := F(i, y')$  where  $\text{Per}(y) = (i, y')$  and the set of all elements queried to  $\mathcal{F}_{\text{opr}}f$  is  $X'$  where  $n' = |X'|$ . The extraction procedure is that  $X = \{x \in \mathbb{F}_p \mid x \in X' \wedge F(x) \in U\}$ . Observe that if there exist two distinct elements  $x_1, x_2 \in X'$  such that  $F(x_1) = F(x_2) \in U$  then more than one element is extracted to  $X$ . The probability of existing collision is  $2^{-v+2\log n'}$  then one approach to avoid collision is choosing  $v = 2\kappa$ . However, when  $v = 2\kappa$ , the overhead communication cost significantly increases.

Therefore, another approach is that Sim only extracts elements  $x \in X'$  if its PRF is distinct and appears in  $U$ , i.e.,  $x \in X'$  such that  $F(x) \in U$  and  $\nexists x' \in X'$  where  $F(x) = F(x')$ . [27] proposed this simulation and claimed that if the output domain of PRF  $v = \kappa$  then this simulation is correct and can not be distinguishable from the real protocol. We point out the proof of [27] has a gap and show that the output of PRF should be  $\kappa + \log n$ .

Indeed, if there exist some  $x_1, x_2 \in X'$  such that  $F(x_1) = F(x_2)$  then Sim only needs to extract  $x_1, x_2$  when one of them is in  $Y$ . Let assume  $x_1 \in Y$ , the probability of  $F(x_2) = F(y)$  for some  $y \in Y$  is  $2^{-v+\log(n_Y)}$  since  $Y$  is first fixed before the function  $F$  is sampled. [27] shows  $n_Y = O(\kappa)$  then the security can hold if  $v = \kappa$ . However, this should be  $v = \kappa + \log n_Y$  since  $n_Y = O(\text{poly}(\kappa))$  instead of  $O(\kappa)$ . In particular, PSI protocols in [27] are targeted on large input set because of the usage of vector OLE.

**Theorem 2.** *The PSI protocol on Fig. 5 securely realizes the ideal functionality  $\mathcal{F}_{\text{psi}}$  over the field  $\mathbb{F}_p$  for set size  $n$  and malicious set size  $n_X = n$ ,  $n_Y = Nd$  with statistical security against malicious adversaries in  $\mathcal{F}_{\text{opr}}f$  hybrid model.*

In general, the malicious PSI (Fig. 5) has a communication cost that depends on the security parameter  $\kappa$  and is dominated by  $\kappa n$ . We now present a new PSI protocol that is secure in malicious setting via a dual execution while its communication cost only depends on the statistic parameter  $\lambda$  and the set size  $n$ . The idea of using a dual execution has been used in [26] but when combining this with our BaRK-OPRF it achieves efficient results, i.e., the total communication cost is only  $2Nd(\ell - \log N) + n(\lambda + \log n) + o(n)$ . The detailed construction of dual PSI is shown in the Appendix of full version [9].

## 5 A Standard PSI from Subfield-Ring OLE

In this section, we describe a new PSI protocol, which builds upon a (simple variant of) a pseudorandom correlation generator for the ring-OLE correlation [7]. Our protocol enjoys a number of important features: it is in the *standard model*, achieves *malicious security* at essentially no cost, has *low communication* (competitive even with the best maliciously secure PSI protocols in the random oracle model), and reasonable computation (albeit superlinear in  $n$ ). Our protocol can also be generalized to a powerful notion of *batch non-interactive PSI*, where (after

a small logarithmic-cost preprocessing step with each server) a client can broadcast a single encoding of his database, and then obtain the intersection with any of the server databases at any time after receiving a single message from this server. We believe that this functionality itself is of independent interest.

### 5.1 Semi-Honest Batch Non-Interactive PSI from Subfield Ring-OLE

We describe a new PSI scheme in the semi-honest model. Our protocol enjoys two interesting features: (1) it is in the standard model, and (2) it is a *batch non-interactive* protocol, a useful communication pattern which we describe afterwards. The full construction is represented on Fig. 6.

**Theorem 3.** *The PSI protocol on Fig. 6 securely realizes the ideal functionality  $\mathcal{F}_{\text{psi}}$  over the field  $\mathbb{F}_p$  with set size  $n$  and malicious set size  $n' = n_X = n_Y = 2n$ , with statistical security against augmented semi-honest adversaries in the  $\mathcal{F}_{\text{sole}}$  hybrid model.*

Above, “augmented semi-honest security” refers to a strengthening of honest-but-curious corruption where the adversary is allowed to change the corrupted parties’ inputs. This is a standard strengthening of semi-honest security, which has been argued to better capture real-world security [16]. It will also facilitate upgrading security to the malicious setting later on.

**Batch Non-interactivity.** To securely realize the functionality  $\mathcal{F}_{\text{sole}}$ , we rely on the PCG-based protocol of [7] (using a straightforward adaptation to the subfield setting), which is secure under the ring-LPN assumption. Interestingly, instantiating the subfield ring OLE this way allows to import a powerful feature of the PCG of [7], which is its *programmability*: when generating a ring-OLE correlation, the receiver can ensure that her output  $a$  remains *identical* across multiple instances of the protocol with different parties.

This feature enables the following communication structure: after a short (logarithmic-communication) interaction with  $N$  servers, a client, playing the role of Alice with input set  $A$ , can broadcast a single compact encoding of her dataset to all the servers (with input sets  $B_1 \cdots B_N$ ). Afterwards, each server  $B_i$  can at any time send a single message  $m_i$  to Alice, from which she can recover  $A \cap B_i$  without further interaction. To our knowledge, this batch non-interactive communication pattern was never achieved by any prior proposal; we believe that it can make our protocol appealing in realistic scenarios.

More concretely, after a logarithmic-communication preprocessing phase where Alice sets up PCG seeds with each of servers, Alice broadcasts the value  $t_A = a - p_A$  to everyone, which communicates  $2n \log p \approx 2\ell n$  bits. This message can be seen as a compact public encoding of her dataset (it is only twice as large as Alice’s set). Afterwards, each server can complete the protocol of Fig. 6 by sending a single message  $(b_1, t_B)$  to the receiver, of length  $3n \log q \approx 3(\lambda + 2 \log n)n$ , from which the receiver can locally recover  $X \cap X_i$ .

Furthermore, using the encoding technique of [31], the  $\lambda + 2 \log n$  term can be reduced to  $\lambda + \log n$  (the improvement is based on the observation that for an appropriate ordering,  $n$  random elements of a set of size  $2^{\lambda+2 \log n}$  are on average at distance  $2^{\lambda+\log n}$  for each other, hence the cost of transmitting them can be reduced to essentially  $\lambda + \log n$  per element by sending the distance between consecutive elements instead).

**Efficiency.** The communication cost of protocol (Fig. 6) is  $n \cdot (2 \log p + 3 \log q) + o(n)$  bits of communication. Here, the size of the subfield  $\mathbb{F}_p$  depends only on the bitsize  $\ell$  of the items in the sets  $A$  and  $B$ , hence we can set  $\log p = \ell$ . As we will see in the analysis,  $\log q$  must be set to  $\log q \approx \lambda + 2 \log n$  to guarantee  $\lambda$  bits of statistical security. This leads to a total communication of  $n \cdot (2\ell + 3\lambda + 6 \log n) + o(n)$  bits, which is reduced to  $n \cdot (2\ell + 3\lambda + 3 \log n) + o(n)$  with the encoding of [31]. The  $o(n)$  term above captures the cost of distributing the PCG seeds of the subfield ring-OLE (we discuss the concrete value of  $o(n)$  later on, for our maliciously secure version of the protocol).

Regarding computation, the computational cost scales as  $O(n \log^2 n)$  due to the fast polynomial interpolations, or as  $O(n \log n)$  when using cyclotomic rings. We provide a concrete analysis of the computational cost of the maliciously secure version of our protocol in Sect. 5.2.

PARAMETERS:

- Two rings  $\mathcal{R}_p = \mathbb{F}_p[X]/F(X) \subseteq \mathcal{R}_q = \mathbb{F}_q[X]/F(X)$ , where  $F(X)$  has degree  $2n + 1$ .
- The sender (Alice) and receiver (Bob) have respective input sets  $A = \{a_1, a_2, \dots, a_n\} \subset \mathbb{F}_p$  and  $B = \{b_1, b_2, \dots, b_n\} \subset \mathbb{F}_p$ .
- A subfield ring-OLE in the ring  $\mathcal{R}_q$  over the subring  $\mathcal{R}_p$ .

PROTOCOL:

1. **(Setting up the correlation)** Alice and Bob encode their sets to  $p_A = \prod_{i=1}^n (X - a_i)$ ,  $p_B = \prod_{i=1}^n (X - b_i)$  respectively, and invoke  $\mathcal{F}_{\text{sole}}$  to generate a subfield ring-OLE correlation over  $\mathcal{R}_p, \mathcal{R}_q$ : Alice receives  $(a, s_A) \in \mathcal{R}_p \times \mathcal{R}_q$  and Bob receives  $(b, s_B) \in \mathcal{R}_q^2$  such that  $s_A + s_B = ab$ .
2. **(Broadcasting the client set encoding)** Alice computes and sends  $t_A = a - p_A$  to Bob.
3. **(Server-to-client message)** Bob sets  $s'_B \leftarrow s_B - t_A b$ . Then, Bob decomposes  $b$  as  $b = b_0 + b_1 \cdot X^n$  (where  $b_0, b_1$  are degree- $n$  polynomials), sets  $s'_B \leftarrow s_B - t_A b$ , and picks a random degree- $n$  polynomial  $b'_0$  over  $\mathcal{R}_q$ . He sends  $b_1$  and  $t_B \leftarrow s'_B + p_B b'_0$  to Alice.
4. **(Output)** Alice sets  $u \leftarrow t_B - p_A b_1 \cdot X^n + s_A$ ; note that  $u = p_A b_0 + p_B b'_0$ . Alice outputs the set  $I = \{x \in A \mid u(x) = 0\}$ .

Fig. 6. Augmented semi-honest PSI protocol based on ring-OLE

### 5.2 Maliciously Secure PSI in the Standard Model

In this section, we upgrade the security of our protocol to the malicious setting. Our upgrade introduces only a minimal communication overhead to the protocol, independent of the set sizes  $n$ . The full protocol is represented on Fig. 7.

**Theorem 4.** *The PSI protocol on Fig. 7 securely realizes the ideal functionality  $\mathcal{F}_{\text{psi}}$  over the field  $\mathbb{F}_p$  with set size  $n$  and malicious set size  $n' = n_X = n_Y = 2n$ , with statistical security against malicious adversaries in the  $\mathcal{F}_{\text{sole}}$ -hybrid model.*

PARAMETERS:

- A field  $\mathbb{F}_{p'}$  and two rings  $\mathcal{R}_p = \mathbb{F}_p[X]/F(X) \subseteq \mathcal{R}_q = \mathbb{F}_{p'}[X]/F(X)$ , where  $F(X)$  has degree  $2n + 1$  and  $|\mathbb{F}_{p'}| \leq |\mathbb{F}_p| - 2$ .  $\text{map}$  is an efficient (and efficiently invertible) injective mapping, with  $\text{map}(\mathbb{F}_{p'}) \subseteq \mathbb{F}_p \setminus \{0, 1\}$ .
- The sender (Alice) and receiver (Bob) have respective input sets  $A = \{a_1, a_2, \dots, a_n\} \subset \mathbb{F}_{p'}$  and  $B = \{b_1, b_2, \dots, b_n\} \subset \mathbb{F}_{p'}$ .
- A subfield ring-OLE in the ring  $\mathcal{R}_q$  over the subring  $\mathcal{R}_p$ .

PROTOCOL:

1. **(Setting up the correlation)** Alice and Bob encode their sets to  $p_A = c \cdot (X - 1) \cdot \prod_{i=1}^n (X - \text{map}(a_i))$  with  $c = -(\prod_{i=1}^n (-\text{map}(a_i)))^{-1}$  (note that this guarantees  $p_A(0) = 1$  and  $p_A(1) = 0$ ) and  $p_B = \prod_{i=1}^n (X - \text{map}(b_i))$  respectively. Alice and Bob invoke  $\mathcal{F}_{\text{sole}}$  to generate a subfield ring-OLE correlation over  $\mathcal{R}_p, \mathcal{R}_q$ : Alice receives  $(a, s_A) \in \mathcal{R}_p \times \mathcal{R}_q$  and Bob receives  $(b, s_B) \in \mathcal{R}_q^2$  such that  $s_A + s_B = ab$ .
2. **(Broadcasting the client set encoding)** Alice computes and sends  $t_A = a - p_A$  to Bob.
3. **(Server-to-client message)** Bob sets  $s'_B \leftarrow s_B - t_A b$ . Then, Bob decomposes  $b$  as  $b = b_0 + b_1 \cdot X^n$  (where  $b_0, b_1$  are degree- $n$  polynomials), and sets  $s'_B \leftarrow s_B - t_A b$ . He sends  $b_1$  to Alice.
4. **(Checking  $p_A$ )** Alice computes  $s'_A \leftarrow s_A - p_A b_1 \cdot X^n$ . Alice sends  $y \leftarrow s'_A(0)$  to Bob. If  $y \neq b_0(0) - s'_B(0)$ , Bob aborts. Else, Bob picks a random degree- $n$  polynomial  $b'_0$  over  $\mathcal{R}_q$  and sends  $t_B \leftarrow s'_B + p_B b'_0$  to Alice.
5. **(Output)** Alice sets  $u \leftarrow t_B - p_A b_1 \cdot X^n + s_A$ ; note that  $u = p_A b_0 + p_B b'_0$ . If  $u(1) = 0$ , Alice aborts; otherwise, Alice computes the set  $I = \{x \in A \mid u(\text{map}(x)) = 0\}$  and outputs  $I$ .

**Fig. 7.** Maliciously secure PSI protocol in the  $\mathcal{F}_{\text{sole}}$ -hybrid model

**Efficiency.** Our malicious protocol has minimal communication overhead over our augmented semi-honest protocol. The main overhead stems from starting from a slightly larger field in which two elements can be “reserved elements”. If  $p'$  is a prime power and  $\ell \approx \log p'$ , the price to pay is therefore increasing  $\ell$  to  $\log p$  where  $p$  is the smallest prime power above  $p' + 2$ . While an exact expression would

be rather tedious, for any reasonable input size this cost should be negligible (the simplest strategy is to pick  $\mathbf{p}' = 2^\ell$  and  $\mathbf{p} = 2^{\ell+1}$ , in which case  $\ell$  is increased by one bit, but much better encoding methods exist). Therefore, the communication remains  $n \cdot (2\ell + 3\lambda + 6 \log n) + o(n)$  bits, or  $n \cdot (2\ell + 3\lambda + 3 \log n) + o(n)$  with the encoding of [31]. We provide a more concrete analysis of the  $o(n)$  term (setting up the ring-OLE) in the malicious setting in the Appendix of full version [9].

*Computation Cost.* Note that our standard model protocol shares with our other protocols the feature of having a communication independent of  $\kappa$ . Our protocol requires more computation compared to the best ROM-based protocols, due to its use of polynomial interpolation. However, it still allows for very fast PSI computation (we estimate a few seconds to compute the intersection between databases of size  $2^{20}$ , on one core of a standard laptop). Concretely, the protocol requires only

- a single degree- $n$  polynomial interpolation, one FFT over a polynomial ring with degree- $2n$  polynomials, and 3 multiplications of degree- $n$  polynomials for the receiver, and
- a single degree- $n$  polynomial interpolation, one FFT as above, 2 multiplications of degree- $n$  polynomials, and a single  $n$ -multipoint polynomial evaluation for the sender.

Furthermore, both polynomial interpolations only have to be performed over a field  $\mathbb{F}$ , of size  $|\mathbb{F}| \approx 2^\ell$  where  $\ell$  is the bit size of the set items (e.g. 32 or 64 bits), and the multipoint evaluation is over a field of size  $\lambda + 2 \log n$  bits. This stands in stark contrasts with previous state of the art protocols [20] that relied on polynomial interpolation (*on top* of using the ROM), where the interpolations and multipoint evaluations had to be performed over a very large field  $\mathbb{F}$  of size  $|\mathbb{F}| \approx 2^{400}$ . By using a cyclotomic ring, the FFTs and polynomial multiplications are much faster than the interpolations. On Table 2, we compare our protocol to the current fastest maliciously secure PSI protocols [21, 27, 29].

**On the attacks of [1].** We note that constructing maliciously secure PSI protocols using an algebraic approach, along the lines of our protocol, is known to be non-trivial and error prone. Indeed, previous works [14] used a similar approach based on polynomial manipulation, OLEs, and the lemmas about the polynomial (appear in the Appendix of full version [9]), to build a malicious PSI protocol. However, their protocol was found to be insecure in a recent preprint, which described powerful concrete attacks on this proposal [1]. Intuitively, the key technical difficulties revolve in both cases around how to handle null polynomials ( $p_A = 0$  or  $p_B = 0$ ). In our specific context, it turns out that our direct use of ring-OLE enables relatively elegant and simple (in hindsight) strategies to enforce nonzero polynomials. Our modification has almost no impact on the communication or the computation of our protocol, essentially giving us malicious security for free (though we note that we still require an additional round of communication). It is not, however, completely clear how to adapt our strategy to the setting of OLE-based algebraic PSI in [14]. We believe that this provides



further support for the intuition that ring-OLE is the right primitive to build PSI protocols using this algebraic approach (beyond its direct advantage in terms of communication efficiency) (Fig. 7).

**Acknowledgement.** We thank all reviewers for their helpful comments. The first author is supported by Dim Math Innov funding from the Paris Mathematical Sciences Foundation (FSMP) funded by the Paris Ile-de-France Region, and the second author acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE). This work was also supported by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute.

## References

1. Abadi, A., Murdoch, S.J., Zacharias, T.: Polynomial representation is tricky: Maliciously secure private set intersection revisited. Cryptology ePrint Archive, Report 2021/1009 (2021). <https://ia.cr/2021/1009>
2. Applebaum, B., Damgård, I., Ishai, Y., Nielsen, M., Zichron, L.: Secure Arithmetic Computation with Constant Computational Overhead. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 223–254. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63688-7\\_8](https://doi.org/10.1007/978-3-319-63688-7_8)
3. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 896–912. ACM Press (Oct 2018)
4. Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019, pp. 291–308. ACM Press (Nov 2019)
5. 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. LNCS, vol. 11694, pp. 489–518. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_16](https://doi.org/10.1007/978-3-030-26954-8_16)
6. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Correlated pseudorandom functions from variable-density LPN. In: 61st FOCS, pp. 1069–1080. IEEE Computer Society Press (Nov 2020)
7. 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, Part II. LNCS, vol. 12171, pp. 387–416. Springer, Heidelberg (Aug 2020)
8. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Orrù, M.: Homomorphic secret sharing: Optimizations and applications. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 2105–2122. ACM Press (Oct / Nov 2017)
9. Bui, D., Couteau, G.: Improved private set intersection for sets with small entries. Cryptology ePrint Archive, Paper 2022/334 (2022). <https://eprint.iacr.org/2022/334><https://eprint.iacr.org/2022/334>
10. Chase, M., Miao, P.: Private Set Intersection in the Internet Setting from Lightweight Oblivious PRF. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12172, pp. 34–63. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-56877-1\\_2](https://doi.org/10.1007/978-3-030-56877-1_2)

11. Couteau, G., Rindal, P., Raghuraman, S.: Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12827, pp. 502–534. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84252-9\\_17](https://doi.org/10.1007/978-3-030-84252-9_17)
12. Dietzfelbinger, M., Weidling, C.: Balanced allocation and dictionaries with tightly packed constant size bins. *Theoret. Comput. Sci.* **380**(1–2), 47–68 (2007)
13. Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious key-value stores and amplification for private set intersection. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12826, pp. 395–425. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84245-1\\_14](https://doi.org/10.1007/978-3-030-84245-1_14)
14. Ghosh, S., Nilges, T.: An algebraic approach to maliciously secure private set intersection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 154–185. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17659-4\\_6](https://doi.org/10.1007/978-3-030-17659-4_6)
15. Ghosh, S., Simkin, M.: The communication complexity of threshold private set intersection. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 3–29. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26951-7\\_1](https://doi.org/10.1007/978-3-030-26951-7_1)
16. Hazay, C., Lindell, Y.: A note on the relation between the definitions of security for semi-honest and malicious adversaries. *Cryptology ePrint Archive*, Report 2010/551 (2010). <https://eprint.iacr.org/2010/551>
17. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005). [https://doi.org/10.1007/11535218\\_15](https://doi.org/10.1007/11535218_15)
18. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 818–829. ACM Press (Oct 2016)
19. Kolesnikov, V., Rosulek, M., Trieu, N., Wang, X.: Scalable private set union from symmetric-key techniques. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11922, pp. 636–666. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34621-8\\_23](https://doi.org/10.1007/978-3-030-34621-8_23)
20. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: SpOT-light: lightweight private set intersection from sparse OT extension. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 401–431. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_13](https://doi.org/10.1007/978-3-030-26954-8_13)
21. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: PSI from PaXoS: Fast, malicious private set intersection. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 739–767. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45724-2\\_25](https://doi.org/10.1007/978-3-030-45724-2_25)
22. Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: Private set intersection using permutation-based hashing. In: Jung, J., Holz, T. (eds.) USENIX Security 2015, pp. 515–530. USENIX Association (Aug 2015)
23. Pinkas, B., Schneider, T., Weinert, C., Wieder, U.: Efficient circuit-based PSI via Cuckoo Hashing. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 125–157. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_5](https://doi.org/10.1007/978-3-319-78372-7_5)
24. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: Fu, K., Jung, J. (eds.) USENIX Security 2014, pp. 797–812. USENIX Association (Aug 2014)

25. Rindal, P., Raghuraman, S.: Blazing fast PSI from improved OKVS and subfield VOLE. *IACR Cryptol. ePrint Arch.* p. 320 (2022). <https://eprint.iacr.org/2022/320>
26. Rindal, P., Rosulek, M.: Malicious-secure private set intersection via dual execution. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) *ACM CCS 2017*, pp. 1229–1242. ACM Press (Oct/Nov 2017)
27. Rindal, P., Schoppmann, P.: VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE. In: Canteaut, A., Standaert, F.-X. (eds.) *EUROCRYPT 2021*. LNCS, vol. 12697, pp. 901–930. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77886-6\\_31](https://doi.org/10.1007/978-3-030-77886-6_31)
28. Rosulek, M., Trieu, N.: Compact and malicious private set intersection for small sets. *Cryptology ePrint Archive, Report 2021/1159* (2021). <https://eprint.iacr.org/2021/1159>
29. Rosulek, M., Trieu, N.: Compact and malicious private set intersection for small sets. *Cryptology ePrint Archive, Report 2021/1159* (2021). <https://ia.cr/2021/1159>
30. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) *ACM CCS 2019*, pp. 1055–1072. ACM Press (Nov 2019)
31. Tamrakar, S., Liu, J., Paverd, A., Ekberg, J.E., Pinkas, B., Asokan, N.: The circle game: Scalable private membership test using trusted hardware. In: Karri, R., Sinanoglu, O., Sadeghi, A.R., Yi, X. (eds.) *ASIACCS 17*, pp. 31–44. ACM Press (Apr 2017)
32. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In: *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1074–1091. IEEE (2021)
33. Wieder, U., et al.: Hashing, load balancing and multiple choice. *Foundations Trends® Theor. Comput. Sci.* **12**(3–4), 275–379 (2017)
34. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated OT with small communication. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) *ACM CCS 20*, pp. 1607–1626. ACM Press (Nov 2020)



# Pseudorandom Correlation Functions from Variable-Density LPN, Revisited

Geoffroy Couteau<sup>1</sup>(✉) and Clément Ducros<sup>2</sup>

<sup>1</sup> CNRS, IRIF, Université de Paris, Paris, France  
couteau@irif.fr

<sup>2</sup> IRIF, INRIA, Université de Paris, Paris, France

**Abstract.** Pseudorandom correlation functions (PCF), introduced in the work of (Boyle *et al.*, FOCS 2020), allow two parties to locally generate, from short correlated keys, a near-unbounded amount of pseudorandom samples from a target correlation. PCF is an extremely appealing primitive in secure computation, where they allow to confine all pre-processing phases of all future computations two parties could want to execute to a single short interaction with low communication and computation, followed solely by offline computations. Beyond introducing the notion, Boyle *et al.* gave a candidate construction, using a new *variable-density* variant of the learning parity with noise (LPN) assumption. Then, to provide support for this new assumption, the authors showed that it provably resists a large class of linear attacks, which captures in particular all known attacks on LPN.

In this work, we revisit the analysis of the VDLPN assumption. We make two key contributions:

- First, we observe that the analysis of Boyle *et al.* is purely asymptotic: they do not lead to any concrete and efficient PCF instantiation within the bounds that offer security guarantees. To improve this state of affairs, we combine a new variant of a VDLPN assumption with an entirely new, much tighter security analysis, which we further tighten using extensive computer simulations to optimize parameters. This way, we manage to obtain for the first time a set of *provable* usable parameters (under a simple combinatorial conjecture which is easy to verify experimentally), leading to a concretely efficient PCF resisting all linear tests.
- Second, we identify a flaw in the security analysis of Boyle *et al.*, which invalidates their proof that VDLPN resists linear attacks. Using several new non-trivial arguments, we repair the proof and fully demonstrate that VDLPN resists linear attack; our new analysis is more involved than the original (flawed) analysis.

Our parameters set leads to PCFs with keys around 3 MB allowing  $\sim 500$  evaluations per second on one core of a standard laptop for 110 bits of security; these numbers can be improved to 350 kB keys and  $\sim 3950$  evaluations/s using a more aggressive all-prefix variant. All numbers are quite tight: only within a factor 3 of the best bounds one could heuristically hope for.

## 1 Introduction

The generation of secret correlated random string is the cornerstone of secure computation (MPC). Given access to a trusted source of correlated randomness, any  $n$ -party functionality can be securely computed with information-theoretic security (against  $n - 1$  corrupted parties), and with very high concrete efficiency. For example, given  $2m$  random oblivious transfers (in a random oblivious transfer, Alice gets two random bits  $(s_0, s_1)$ , and Bob gets  $(b, s_b)$  for a random bit  $b$ ), two parties can securely compute any boolean circuit  $C$  with up to  $m$  AND gates, with perfect security, while exchanging only four bits per AND gate.

The simplicity and efficiency of this paradigm is well known, and most modern MPC protocols take advantage of its features by sharing the same high level two-step structure: in the first step, the *preprocessing phase*, the parties interact to distributively and securely generate these correlated randomness. Since this phase is input-independent, it can be carried out ahead of time. Then, in the second step, the *online phase*, the parties “consume” this correlated randomness in a fast, information-theoretic protocol. The core challenge is this approach lies in step 1: designing a secure protocol to distributively generate long correlated random string.

**Pseudorandom Correlations.** Until recently, all state of the art protocols, such as SPDZ [DPSZ12], required  $\Omega(s)$  communication to generate  $s$  bits of correlated randomness (ignoring terms depending on the security parameter and the number of parties), leading to communication-intensive preprocessing phases. This state of affair changed in a recent and exciting line of work [BCG+17, BCGI18, BCG+19b, BCG+19a, SGRR19, BCG+20b, CRR21] which introduced the notion of *pseudorandom correlation generators* (PCG), a new cryptographic primitive which allows parties to locally generate, from short correlated seeds, long instances of correlated *pseudorandom* strings. These PCGs enable secure computation with *silent preprocessing* where, after a short interaction to generate the short correlated seeds, the parties never need to interact anymore, and locally generate the long correlated strings. The latest results in this area further demonstrated that this primitive could be achieved with very high concrete efficiency, under appropriate LPN-like cryptographic assumptions.

**Pseudorandom Correlated Functions.** The aforementioned constructions of PCG, however, share a common limitation: the expansion of the short keys into long pseudorandom correlated strings is a one-time, monolithic procedure. That is, these PCGs are limited to a single generation of an a priori bounded amount of correlated pseudorandomness. If the parties want to possibly use these correlations across many protocols, then they carry the burden of having to either re-do the distributed generation of the short keys each time, or storing a very large amount of correlated randomness for a possibly long duration.

These limitations were overcome in a recent work [BCG+20a], where the authors introduced the notion of pseudorandom correlated *functions*

(PCFs). PCFs are to PCGs what pseudorandom functions are to pseudorandom generators: they allow to generate an arbitrary amount of correlated (pseudo)randomness in an incremental fashion. That is, given two short correlated keys  $(K_0, K_1)$ , two parties can locally compute an arbitrary number of correlated strings  $F_{K_0}(x), F_{K_1}(x)$ , which are all indistinguishable from independent random samples from the target correlation. PCFs allow to confine *all future preprocessing phases* of any future MPC protocols that two parties may wish to run to a *one-time* short interaction, followed solely by local computation to generate the preprocessing material in all subsequent computations.

## 1.1 Constructions of Pseudorandom Correlated Functions

A PCF is an extremely powerful primitive, but also one which is highly non-trivial to construct. A generic construction of PCF under the LWE assumption can be obtained by letting the two parties homomorphically evaluate a well-chosen circuit using a threshold fully-homomorphic encryption scheme [DHRW16, BCG+20a]: the circuit takes as input a PRF key  $K$ , and computes pseudorandom instances of the correlation, using the output of the PRF to generate the pseudorandomness used in these correlations. However, this approach falls short of providing a concretely usable solution. To our knowledge, there are currently two competing approaches to construct usable PCFs:

**PCFs from Variable-Density LPN.** The work of [BCG+20a] gave a generic construction of PCF, by combining two primitives:

- A function secret sharing scheme (FSS) for a class of circuits  $\mathcal{C}$ . At a high level, an FSS for  $\mathcal{C}$  allows to share any function  $f \in \mathcal{C}$  in two functions  $f_0, f_1$  such that each  $f_i$  computationally hides  $f$ , yet for any input  $x$ , it holds that  $f_0(x) + f_1(x) = f(x)$ .
- One weak pseudorandom function (WPRF) for some class  $\mathcal{C}'$  related to  $\mathcal{C}$ . A WPRF is a PRF where the adversary in the pseudorandomness game is restricted to only querying random inputs.

Previous works [GI14, BGI15, BGI16] have shown how to construct extremely efficient FSS schemes for simple complexity classes, such as multi-point functions (i.e., a function  $f_{\alpha, \beta}$  equal to 0 everywhere, except on  $n$  specific points  $\alpha = (\alpha_1, \dots, \alpha_n)$ , where it takes a fixed value  $\beta$ ), from minimal assumptions (namely, the existence of one-way functions). The shares of an  $n$ -point function  $f_{\alpha, \beta}$  over a domain of size  $N$  consist of  $n \log N$  PRG seeds, and evaluating  $f_i$  on the entire domain requires only  $N$  PRG evaluations. Given this, the authors of [BCG+20a] put forward a new WPRF in the (particularly low) complexity class of multi-point functions, which essentially boils down to a WPRF of the form  $F_K(x) = F(x \oplus K)$ , where  $F$  is a depth-two circuit with one bottom layer of high fan-in ANDs, and a single top high fan-in XOR gate. The security of this new candidate relies on the hardness of a new variant of the learning parity with noise (LPN) assumption, called *variable density* LPN assumption; we will

overview this assumption later on. Given this new WPRF and the efficient FSS scheme of [BGI16], the authors of [BCG+20a] obtain a PCF candidate which can handle a wide variety of low-degree correlations, including (but not limited to) oblivious transfer correlations. The authors provide several variants and parameter choices; their most aggressive choices of parameters lead to a reasonably efficient construction, which (based on rough estimations) could generate hundreds to thousands of pseudorandom OT correlations per second on one core of a standard computer.

**PCFs from Decisional Composite Residuosity.** An alternative approach to building PCFs was recently put forward in [OSY21], using a Paillier-based construction of homomorphic secret sharing. In contrast to [BCG+20a], this work does not need to rely on new assumptions, and instead only requires the well-established decision composite residuosity assumption. However, this alternative construction has several downsides:

- *Expressivity.* The construction of [OSY21] is inherently limited to oblivious transfer correlations. In contrast, the VDLPN-based construction can generate arbitrary low-degree polynomial correlations, such as OLE, (authenticated) Beaver triples, and many more; these alternative correlations are crucial in many secure computation protocols.
- *Post-quantumness.* The DCR assumption can be broken by Shor’s algorithm. In contrast, while VDLPN is a new and little studied assumption, there seems to be no reason to believe that it should be quantumly broken, being a relatively natural LPN-style assumption.
- *Efficiency.* Eventually, the construction of [OSY21] requires a few hundred exponentiations in an RSA group for every OT correlation produced. Using standard benchmark for exponentiations in 2048-bit RSA groups on a modern laptop<sup>1</sup>, this translates to a cost of the order of one second *for each OT produced*, which is several orders of magnitude less efficient than what the VDLPN-based approach can plausibly provide, for suitable choices of parameters.

Given the above, the VDLPN approach seems to provide the best alternative to obtain efficient and expressive PCFs; however, its reliance on a new assumption calls for a very careful examination of its security. The work of [BCG+20a] provided an initial security analysis, proving a number of important results regarding the resistance of VDLPN against standard attacks. However, this analysis is purely asymptotic, and does not say much about what concrete choices of parameters can be expected to provide a sufficient security level. In addition, a close inspection of their analysis uncovers an important gap in one of the claim, invalidating part of the analysis (we will expand on this later on). Before we detail our contribution, we provide more context on the underlying new assumption and its analysis.

<sup>1</sup> E.g. A laptop equipped with an Intel i5 2540M processor can compute an RSA decryption in 1.4ms of amortized time.

## 1.2 The Variable-Density LPN Assumption

At a high level, the standard LPN assumption with dimension  $k$  and number of samples  $n > k$  states the following: given a uniformly random matrix  $A \xleftarrow{\$} \mathbb{F}_2^{n \times k}$ , sample a vector  $\mathbf{b}$  as  $\mathbf{b} = A \cdot \mathbf{s} + \mathbf{e}$ , where  $\mathbf{s}$  is a random vector from  $\mathbb{F}_2^k$ , and  $\mathbf{e}$  is a random *sparse* vector (the noise vector) over  $\mathbb{F}_2^n$  (the exact distribution of  $\mathbf{e}$  depends on the LPN flavor: it follows a Bernoulli distribution for standard LPN, it is uniform over all vectors of a given weight for exact LPN (XLPN), and it is a concatenation of unit vectors for regular LPN). Then, LPN states that it is hard to distinguish  $\mathbf{b}$  from a uniformly random vector (put otherwise, it is hard to solve noisy systems of linear equations).

In coding theoretic terms, LPN therefore states that a noisy codeword from a random linear code looks random. LPN admits an equivalent, dual formulation: viewing  $A$  as the generating matrix of a linear code of dimension  $k$ , let  $H \in \mathbb{F}_2^{(n-k) \times n}$  be a *parity-check matrix* of  $A$  (which satisfies  $H \cdot A = 0$ ; that is,  $H^\top$  generates the dual of the code generated by  $A$ ). Then distinguish  $\mathbf{b} = A \cdot \mathbf{s} + \mathbf{e}$  from random is equivalent to distinguishing  $H \cdot \mathbf{b} = H \cdot \mathbf{e}$  from random – that is, finding whether an undetermined system of linear equation admits a sparse solution. This is also known as the *syndrome decoding* problem.

The (dual) LPN assumption implies a natural construction of pseudorandom generators, which maps (a short description of)  $\mathbf{e}$  to  $H \cdot \mathbf{e}$ . This PRG (and variants thereof) is at the heart of all known construction of pseudorandom *correlation* generators, due to its linear structure which allows to preserve some target correlations. To obtain a pseudorandom correlation *function*, the work of [BCG+20a] faced the following dilemma: intuitively, we would like to extend the PRG that maps (a representation of)  $\mathbf{e}$  to  $H \cdot \mathbf{e}$  into a PRF, but this means that we need  $H \cdot \mathbf{e}$  to be an exponentially long vector whose entry can be generated incrementally (in this view, an input defines a row  $\mathbf{h}$  of the matrix  $H$ , the key defines  $\mathbf{e}$ , and the corresponding output is  $\mathbf{h}^\top \cdot \mathbf{e}$ ). We need a way to guarantee that  $\mathbf{e}$  and the rows of  $H$  both admit a short (polynomial size) representation, and that  $\mathbf{h}^\top \cdot \mathbf{e}$  can be computed in polynomial time. Unfortunately, defining  $H$  and  $\mathbf{e}$  to be exponentially sparse does not work in general:  $H \cdot \mathbf{e}$  would then become sparse as well, and therefore trivial to distinguish from random.

The key observation in [BCG+20a], and the central idea of their design, is that we can circumvent this issue by making  $H$  and  $\mathbf{e}$  exponentially sparse, but with *variable density*. Concretely, fix a security parameter  $\lambda$  and consider sampling the rows of  $H$  as follows: a row  $\mathbf{h}$  is divided into  $\lambda$  blocks  $(\mathbf{h}_i)_{i \leq \lambda}$  (looking ahead, the maximum number of queries to the PRF will be bounded by a quantity smaller than  $2^\lambda$ ). Each block  $\mathbf{h}_i$  is of length  $\lambda \cdot 2^i$  and contains exactly  $\lambda$  1's: this guarantees that the *density* of  $\mathbf{h}_i$  is  $1/2^i$ . More precisely,  $\mathbf{h}_i$  is a concatenation of  $\lambda$  length- $2^i$  unit vectors. This means that  $\mathbf{h}$  constructed this way is a *variable density* vector, where the density drops by a factor two when going from one block to the next. The noise vector  $\mathbf{e}$  is simply sampled as a row vector. Intuitively, the dense portion of the inner products  $\mathbf{h}^\top \cdot \mathbf{e}$  guarantees that the result will not be sparse (but the corresponding portions being narrow, many linear dependencies appear), while the sparse portions of the  $\mathbf{h}^\top \cdot \mathbf{e}$  break



linear dependencies (being exponentially wide, though very sparse). The VDLPN assumption states, informally, that this suffices to guarantee indistinguishability from random.

**Definition 1 (VDLPN assumption, informal).** *Sample a matrix  $H$  as  $H = H_1 || \dots || H_\lambda$  over  $\mathbb{F}_2^{N \times \lambda \cdot (2^{\lambda+1} - 1)}$  where the rows are independently sampled as described above, and where  $N \ll 2^\lambda$  is some bound on the maximum number of queries. Sample a noise vector  $\mathbf{e}$  according to the same distribution as the rows of  $H$ . Then the VDLPN assumption states that, given  $H$ ,  $H \cdot \mathbf{e}$  is indistinguishable from a random length- $N$  vector.*

VDLPN directly implies a natural construction of WPRF: a random input  $x$  (a bitstring of length  $\lambda^2 \cdot (\lambda + 1) / 2$ ) is parsed as  $\lambda$  blocks of length  $\lambda \cdot i$ , for  $i = 1$  to  $\lambda$ , where each block is further parsed as  $\lambda$  sub-blocks of length  $i$  each. A length- $i$  string defines a random unit vector of length  $2^i$  (it encodes the position of the nonzero entry in the vector). The concatenation of these unit vectors forms a uniformly random row  $\mathbf{h}_x$  for the matrix  $H$ . A similar mapping is applied to convert the bitstring  $K$  (the WPRF key) into a noise vector  $\mathbf{e}_K$ . Eventually, observe that the mapping  $F_K : x \rightarrow \mathbf{h}_x^\top \cdot \mathbf{e}_K$  is efficient, because each of  $\mathbf{h}_x \cdot \mathbf{e}_K$  is exponentially sparse: computing their inner product amounts to computing  $O(\lambda^2)$  equality tests between the sub-blocks of  $x$  and of  $K$ . To construct a pseudorandom *correlation* function, the authors of [BCG+20a] build upon the fact that this WPRF can further be written as a XOR of point functions (each point function takes a sub-block of  $x$  as input and returns 0 unless it is equal to the corresponding sub-block of  $K$ ), which makes it *FSS-friendly*.

### 1.3 Security of VDLPN

Since VDLPN is a variant of the LPN assumption, the natural first step to analyze its security is to look at existing attacks on LPN. There have been, however, a tremendous number of attacks on LPN designed over the years, including attacks such as Gaussian elimination and the BKW algorithm [BKW00, Lyu05, LF06, EKM17] and variants based on covering codes [ZJW16, BV16, BTV16, GJL20], and attacks based on information set decoding techniques [Pra62, Ste88, FS09, BLP11, MMT11, BJMM12, MO15, EKM17, BM18]. This list is far from exhaustive; one could also mention statistical decoding attacks [AJ01, FKI06, Ove06, DAT17], generalized birthday attacks [Wag02, Kir11], linearization attacks [BM97, Saa07], attacks based on finding low weight code vectors [Zic17], and many more. A core observation of [BCG+20a] is that *all* these attacks fit in a common framework, called *linear tests*. Roughly, a linear test is an attack in which the adversary attempts to distinguish  $\mathbf{b}$  from a random vector by finding a nonzero linear function  $L_H$  (which can depend on  $H$  in an arbitrary way) such that  $L_H(\mathbf{b})$  is *biased* (i.e., far from uniform in statistical distance) when  $\mathbf{b} = H \cdot \mathbf{e}$ . Being secure against linear tests is a statistical property, which one can hope to prove unconditionally. To this end, the work of [BCG+20a] put forward the notion of *low bias* WPRF:

**Definition 2 (low-bias WPRF, informal).** A family  $\{F_K\}_K$  of WPRFs has low bias up to  $N$  samples if

$$\Pr_{x_1, \dots, x_N} [\text{bias}(\mathcal{D}) \geq \text{negl}(\lambda)] \leq \text{negl}'(\lambda),$$

where  $\mathcal{D}$  is the distribution that samples a random key  $K$  for the WPRF, and outputs the vector  $(F_K(x_1), \dots, F_K(x_N))$ . Above, the  $N$  inputs are sampled from the input space of the WPRF family, and  $\text{negl}, \text{negl}'$  denote two negligible functions.

Above, the bias of a distribution  $\mathcal{D}$  over  $\mathbb{F}_2^N$  is defined as  $\max_{\mathbf{u} \neq \mathbf{0}} |1/2 - \Pr_{\mathbf{v} \leftarrow \mathcal{D}}[\mathbf{u}^\top \cdot \mathbf{v} = 1]|$ ; that is, it is the distance from the uniform distribution over  $\mathbb{F}_2$  induced by computing  $L(\mathbf{v})$  with the “worst possible” nonzero linear function  $L : \mathbb{F}_2^N \mapsto \mathbb{F}_2$ . One of the core security claims of [BCG+20a] hinges upon the fact that the VDLPN-based WPRF is a low-bias WPRF; in particular, this means that the VDLPN assumption cannot be broken using essentially any of the known attacks on LPN.

**Theorem 3 (Resistance to linear tests [BCG+20a], informal).** The WPRF built from the VDLPN assumption has a low bias up to  $N = 2^{O(\lambda)}$  samples (the functions  $\text{negl}, \text{negl}'$  are both equal to  $2^{O(-\lambda)}$  as well).

To show that the VDLPN assumption is secure, we will only consider resistance against linear tests - and all our proof of security will consist of showing this resistance. A simple variant of the VDLPN assumption achieves smaller input size ( $O(\lambda^2)$  instead of  $O(\lambda^3)$ ), but we ignore it in this simplified overview. Note that [BCG+20a] also considers various other attacks, such as algebraic attacks, linear cryptanalysis, and attacks by low depth ( $\text{AC}^0$ ) circuits. These analyses make VDLPN a plausible assumption, from which [BCG+20a] derives several consequences: a pseudorandom correlation function, as we already discussed, but also the first candidate WPRF in the very low complexity class XOR-AND (one layer of ANDs followed by a single XOR gate), which indicates that this class is perhaps hard to learn in the uniform PAC model. Furthermore, VDLPN also implies a WPRF secure against XOR related-key attacks, something which was previously known only assuming very strong cryptographic primitives (namely, high degree multilinear maps).

## 1.4 Our Contributions

We revisit the security analysis of VDLPN against linear tests. Our main motivation is that the analysis in [BCG+20a] is purely asymptotic, and trying to extract concrete parameters within the range where the analysis applies gives terrible performances. Concretely, let  $D$  be the number of blocks,  $w$  be the number of ones in each block, and  $N$  be a bound on the maximum number of queries (in our simplified exposition above, we used  $w = D = \lambda$  and  $N \ll 2^\lambda$ ). The authors of [BCG+20a] suggested the following concrete parameters to instantiate VDLPN: set  $w = 1.5\lambda$ ,  $D = w/4$ , and  $N = 2^D$ . They conjectured that this

should achieve  $\lambda$  bits of security. However, this choice of parameters is purely heuristic, and described as a challenge to cryptanalysts: it is not backed up by any concrete cryptanalysis.

On the other hand, their analysis guarantees  $2^{\Omega(w)}$  bits of security against linear tests whenever  $w > \Gamma \cdot D$ , for up to  $2^D$  samples, where  $\Gamma$  is a constant from the proof. A quick back-of-the-envelope calculation reveals that  $\Gamma$  in their analysis is of the order of magnitude of  $10^5$ , and it is far from obvious to improve the constants without significantly changing the analysis (while the proof is not tight, a straightforward “tightening” only saves a small factor). This means that, when instantiating the parameters within the range where the proof offers some security guarantees, the security parameters must be of the order of *several million bits long* – of course, this is entirely impractical.

Furthermore, upon revisiting their analysis, we uncovered one mistake (as well as a second, relatively minor mistake), that invalidates their proof of security against linear attacks. Fixing the mistake turns out to be non-trivial, and constitutes an important part of our contribution.

**First Contribution: A Tighter VDLPN.** The corrected analysis we present offers even worse concrete bounds than in the original (flawed) proof: the  $\Gamma$  value is of the order of  $10^6$ , leading to a security parameter  $w$  in the millions. In other words, there is no realistically usable range of parameters within the bounds handled by the security analysis. Thus, one is left with a plausible assumption with purely asymptotic parameters on the one hand, and some concrete candidate choices of parameters that lead to a reasonably efficient PCF construction, but that are not supported by any security analysis. The goal of this first contribution is to bridge this (huge) gap between *secure in theory* and *usable in practice*. Since the task is highly non-trivial, we attack the problem simultaneously on three angles. Each angle in itself forms an orthogonal contribution to the overall analysis (in the sense that each of the three techniques leads to significant improvements by themselves).

- *An entirely new proof approach.* First, we step back from the original analysis and seek to understand the main source of slackness in the parameters. Then, we develop an alternative, much more direct approach which, in a sense, allows us to exploit the contribution to the bias of every component of the matrix  $H$  (while the previous analysis could only take into account the contribution of the “top contributors”, for technical reasons). The new approach achieves much tighter bounds.
- *A proof-friendly VDLPN variant.* Second, we allow ourselves to (slightly) *change* the VDLPN assumption. Concretely, our variant is identical to VDLPN, except for the first block  $H_1$ : here, we set  $H_1$  to be a uniformly random matrix instead. This choice stems from the fact that in the analysis, we need to use two different arguments to handle the low weight linear tests and the high weight linear tests; sampling  $H_1$  uniformly at random allows to achieve much tighter bounds for the analysis against low weight tests. We

observe that this variant of VDLPN remains FSS-friendly: using this variant does not harm any of the cryptographic applications.

- *Better bounds through simulations.* Eventually, we rely on extensive computer simulations to achieve tighter bounds. Concretely, we need a bound on the expectation of some complex random variable  $X$ , which we obtained using a generalized Chernoff inequality in the previous analysis. While this bound suffices for the asymptotic analysis, its looseness severely impacts the bounds. Here, instead, we estimate  $\mathbb{E}[X]$  through computer simulations. We empirically observe that the samples from  $X$  have very low variance, and derive a tight bound on  $\mathbb{E}[X]$  with a very high confidence interval.

Putting everything together, we manage to prove that (our variant of) VDLPN has bias at most  $2^{-80}$  with probability at least  $1 - 2^{-80}$ , for a value of  $w$  as low as  $w = 380$  (with  $D = 30$ , and up to  $N = 2^D$  samples – this is just a sample of candidate parameters, we do not have a closed-form formula).<sup>2</sup> This is a tremendous improvement compared to the previous analysis, and gives for the first time a set of parameters which are simultaneously backed by a thorough security analysis, yet are usable in practice. We stress that, in spite of our computer-verified component, our bounds are much better than purely heuristic bounds: they are provable bounds under a simple, concrete combinatorial conjecture, which is easy to verify through computer experiments. In contrast, even ignoring the flaw in their asymptotic analysis, all “usable parameters” proposed in [BCG+20a] were purely heuristic, based on the intuition that they might be hard to attack and described as challenges for cryptanalysis, but not supported by any analysis whatsoever.

We believe that our work constitutes a strong step in the direction of showing that one can construct secure and concretely efficient pseudorandom correlation functions, an important and intriguing goal.

**Second Contribution: Fixing the Original Analysis.** In essence, the analysis of a central claim in the proof of resistance against linear tests turned out to be incorrect. The claim, on the other hand, remains essentially correct (up to some concrete choice of the constants involved): only its analysis is flawed, it did not lead to attacks. The mistake appears in a bound on the expectation  $\mathbb{E}[Z]$  of a random variable  $Z$ , of the form  $\mathbb{E}[Z] \in [a, b]$ , for some values  $0 < a < b$ . The authors deduced from this bound a bound of the form  $\mathbb{E}[|Z - b|] \leq b - a$ , but this is wrong in general (the error might stem from an application of the Jensen inequality in the wrong direction): intuitively, if the distribution of  $Z$  is “anti-concentrated” with respect to its expectation, then the inequality  $\mathbb{E}[|Z - b|] \leq b - a$  does not follow from  $\mathbb{E}[Z] \in [a, b]$ <sup>3</sup>.

<sup>2</sup> The choice of 80 bits of security is more conservative than it appears: it means that an adversary will have to compute  $2^{80}$  inner products with a length- $2^{30}$  vector to detect a  $2^{-80}$  bias in the output. In terms of bit-security, this corresponds to at least 110 bits of security.

<sup>3</sup> E.g. if  $Z$  is 0 with probability  $1/2$ , and 10 else, then  $\mathbb{E}[Z] = 5 \in [4, 5]$ , but  $\mathbb{E}[|Z - 5|] = 5 > 5 - 4$ .

On the other hand, if  $Z$  follows a “nice” distribution, typically a Gaussian-style distribution (or any bell-shape distribution), and if the value  $b$  is sufficiently close to  $\mathbb{E}[Z]$ , then the claim becomes true. A quick simulation reveals that  $Z$  indeed appears to exhibit the right structure. Central to our first contribution is a formal proof that the claim holds for  $Z$ . Compared to the analysis of [BCG+20a], our new analysis cannot simply bound the expected value of  $Z$ : we have to prove strong *tail bounds* on  $Z$ , which is significantly more complex, because  $Z$  is a sum of *dependent* variables. Our analysis relies on a power-full bound about the balls and bins problem.

We then turn to integrating our new proof of the central claim to the full proof of resistance against linear tests. Along the way, we found (and fixed) another minor mistake in the analysis, which requires changing the concrete choices of constants in the proof. Due to this, and due to some slackness in our new proof of the central claim (which stems from the limitations of the inequality which we use), the general proof ends up failing on some corner cases. Essentially, the analysis studies separately the contribution of each block  $H_i$  of the matrix  $H$  to the overall bias; the analysis, however, fails whenever  $i$  is too small. Nevertheless, we show that the case of very small values of  $i$  can be treated separately with two simple arguments, which completes the proof.

We stress that while the repaired proof follows the high level structure of that of [BCG+20a], the core of correction was not straight forward. This security analysis against linear tests is central to the claim that VDLPN is a plausible assumption (since it resists all known attacks against LPN), and therefore provides a plausible candidate to construct powerful objects such as a PCF (for all low-degree correlations), a XOR-RKA secure WPRF, and a family of extremely simple functions (in the XOR-AND class) hard to learn in the uniform PAC model. We also mention that we notified the authors of [BCG+20a] of our findings, and they acknowledged the flaws in the analysis.

## 1.5 New Cost Estimations for PCFs, and Challenges

Using the parameters from above ( $w = 380, D = 30$ ), we compute the seed size and estimate the evaluation time of the pseudorandom correlation function of [BCG+20a] instantiated using our new VDLPN variant. On top of the VDLPN variant, the construction uses a puncturable pseudorandom function, instantiated with the GGM construction [GGM86]. We set the security parameter of the PRG used in GGM to  $\lambda = 128$ . With these parameters, we get the following costs:

- Seed size: 2.94 MB
- PCF evaluation time: the evaluation cost is (largely) dominated by  $\approx 1.81 \cdot 10^5$  calls to a length-doubling pseudorandom generator.

To give a rough runtime estimation, the PRG can be instantiated using two calls to fixed-key AES. According to [MSY21], using the AES-NI instructions of modern CPUs, one byte of AES-128 can be computed in  $\sim 1.3$  cycles. Hence, computing  $3.6 \cdot 10^5$  blocks of 16 bytes requires about  $7.5 \cdot 10^6$  cycles. Concretely,

using a 3.8GHz processor, this amounts to roughly 500 PCF evaluations per second on a single core (note that the estimation should not be too far off, because the computation requires no random data access, hence cache misses are unlikely). Since all evaluations are fully parallelizable, using  $c$  cores increases this number to  $500c$  evaluations per second.

The work of [BCG+20a] also suggested an improved *all prefix* variant, which has shorter seeds and better runtimes, using existing efficient constructions of all-prefix function secret sharing. While this construction lacks a security analysis, this is only because it makes the noise vectors  $\mathbf{e}_i$  correlated (our analysis fundamentally uses their independence). However, it seems very reasonable to conjecture that this is just an artefact of the analysis, and that the optimized construction provides the same security level. Under the heuristic assumption that the correlated  $\mathbf{e}_i$  behave essentially as well as independent  $\mathbf{e}_i$  for resistance to linear tests, we can reuse our previous analysis and obtain the following improved bounds for the *all-prefix* PCF: seed size 0.35MB, and PCF evaluation time around 3950 evaluations per second on a single 3.8GHz processor.

These numbers demonstrate that, already within the range of our provable bounds, PCFs can achieve very promising parameters, with short seeds, and reasonably fast runtimes. Note that we believe that there remains some small gap between our analysis and the “true” security of VDLPN – namely, smaller parameters might plausibly lead to a secure instance (perhaps as small as  $w = 120$  and  $D = 30$ ). We view further tightening our analysis as an interesting open question. Since the cost is linear in  $w$ , reducing  $w$  to 120 would lead to a factor 3 improvement (on seed size and evaluations per second). Nonetheless, our provable parameters appear already quite tight, being at most a factor-3 off compared to the best parameters one could heuristically hope for.

## 2 Preliminaries

We use bold font for vectors, and capitals for matrices. For vectors  $\mathbf{u}, \mathbf{v}$ ,  $\text{HW}(\mathbf{u})$  denotes the Hamming weight of  $\mathbf{u}$ ,  $d_H(\mathbf{u}, \mathbf{v})$  denotes the Hamming distance between  $\mathbf{u}, \mathbf{v}$ . Below, we recall the definition of the bias of a distribution, and some standard technical lemmas.

**Definition 4 (Bias of a Distribution).** *Given a distribution  $\mathcal{D}$  over  $\mathbb{F}^n$  and a vector  $\mathbf{u} \in \mathbb{F}^n$ , the bias of  $\mathcal{D}$  with respect to  $\mathbf{u}$ , denoted  $\text{bias}_{\mathbf{u}}(\mathcal{D})$ , is equal to*

$$\text{bias}_{\mathbf{u}}(\mathcal{D}) = |\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[\mathbf{u}^T \cdot \mathbf{x}] - \mathbb{E}_{\mathbf{x} \sim U_n}[\mathbf{u}^T \cdot \mathbf{x}]| = \left| \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[\mathbf{u}^T \cdot \mathbf{x}] - \frac{1}{|\mathbb{F}|} \right|,$$

where  $U_n$  denotes the uniform distribution over  $\mathbb{F}^n$ . The bias of  $\mathcal{D}$ , denoted  $\text{bias}(\mathcal{D})$ , is the maximum bias of  $\mathcal{D}$  with respect to any nonzero vector  $\mathbf{u}$ .

**Standard Probability Lemmas.** 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  $\mathbf{v}_i \stackrel{\$}{\leftarrow} \mathcal{D}_i$  for  $i = 1$  to  $t$  and outputting  $\mathbf{v} \leftarrow \mathbf{v}_1 \oplus \dots \oplus \mathbf{v}_t$ . We will use the following bias of the exclusive-or (cf. [Shp09]).

**Lemma 5.** *Let  $t \in \mathbb{N}$  be an integer, and let  $(\mathcal{D}_1, \dots, \mathcal{D}_t)$  be  $t$  independent distributions over  $\mathbb{F}_2^n$ . Then  $\text{bias}(\bigoplus_{i \leq t} \mathcal{D}_i) \leq 2^{t-1} \cdot \prod_{i=1}^t \text{bias}(\mathcal{D}_i) \leq \min_{i \leq t} \text{bias}(\mathcal{D}_i)$ .*

Let  $\text{Ber}_r(\mathbb{F}_2)$  denote the Bernoulli distribution that outputs 1 with probability  $r$ , and 0 otherwise. More generally, we denote by  $\text{Ber}_r(\mathbb{F})$  the distribution that outputs a uniformly random element of  $\mathbb{F}$  with probability  $r$ , and 0 otherwise (this does not exactly match our definition of  $\text{Ber}(\mathbb{F}_2)$ , but the slight discrepancy will not matter in our applications). We will use a standard simple lemma for computing the bias of a XOR of Bernoulli samples:

**Lemma 6 (Piling-up lemma).** *For any  $0 < r < 1/2$  and any integer  $n$ , given  $n$  random variables  $X_1, \dots, X_n$  i.i.d. to  $\text{Ber}_r(\mathbb{F}_2)$ , it holds that  $\Pr[\bigoplus_{i=1}^n X_i = 0] = 1/2 + (1 - 2r)^n / 2$ .*

We will also need two concentration bounds. The bounded difference inequality [McD89] is an application of the more general Azuma inequality [Azu67]. Let  $(n, m) \in \mathbb{N}^2$  be two integers. We say that a function  $\Phi : [n]^m \mapsto \mathbb{R}$  satisfies the Lipschitz property with constant  $d$  if for every  $\mathbf{x}, \mathbf{x}' \in [n]^m$  which differ in a single coordinate, it holds that  $|\Phi(\mathbf{x}) - \Phi(\mathbf{x}')| \leq d$ .

**Lemma 7 (Bounded Difference Inequality).** *Let  $\Phi : [n]^m \mapsto \mathbb{R}$  be a function satisfying the Lipschitz property with constant  $d$ , and let  $(X_1, \dots, X_m)$  be independent random variables over  $[n]$ . Then*

$$\Pr[\Phi(X_1, \dots, X_m) < \mathbb{E}[\Phi(X_1, \dots, X_m)] - t] \leq \exp\left(-\frac{2t^2}{m \cdot d^2}\right).$$

Eventually we will rely on the Occupancy Bound from [KMPS94], which provides tight bounds for the balls and bins problem.

**Lemma 8 (Occupancy Bound).** *Let  $E$  be the number of empty bins when  $m$  balls are placed randomly into  $n$  bins, and define  $r = m/n$ . The expectation of  $E$  is given by  $\mu = \mathbb{E}[E] = (1 - \frac{1}{n})^m \approx ne^{-r}$ . For any  $\theta > 0$ ,*

$$\Pr[|E - \mu| \geq \theta\mu] \leq 2 \exp\left(-\frac{\theta^2 \mu^2 (n - \frac{1}{2})}{n^2 - \mu^2}\right) = \mathcal{B}$$

*Note that we can derive the following two equations :  $\Pr[E \geq \mu(\theta + 1)] < \mathcal{B}$  and  $\Pr[E \leq \mu(1 - \theta)] < \mathcal{B}$ .*

### 2.1 Coding Theory

**Definition 9.** *Let  $n$  be a positive integer,  $\mathcal{C}$  is a linear code if  $\mathcal{C}$  is a vector subspace of  $\mathbb{F}_q^n$ . The integer  $n$  is called the length of  $\mathcal{C}$ . The dimension of  $\mathcal{C}$  is its dimension as an  $\mathbb{F}_q$ -vector space. It is denoted by  $k = \dim_{\mathbb{F}_q} \mathcal{C}$*

**Definition 10.** *(Minimum distance of a code) Let  $\mathcal{C}$  be a linear code of length  $n$ . The minimum distance of  $\mathcal{C}$ , is the minimum distance  $d_C$  between two distinct codewords of  $\mathcal{C}$ .*

$$d_C = \min_{\mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}} \{d_H(\mathbf{u}, \mathbf{v})(\mathbf{x}, \mathbf{y})\}$$

**Learning Parity with Noise.** We define the LPN assumption over a ring  $\mathcal{R}$  with dimension  $k$ , number of samples  $n$ , w.r.t. a code generation algorithm  $\mathbf{C}$ , and a noise distribution  $\mathcal{D}$ :

**Definition 11 (Dual LPN).** Let  $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{k,n}(\mathcal{R})\}_{k,n \in \mathbb{N}}$  denote a family of efficiently sampleable distributions over a ring  $\mathcal{R}$ , such that for any  $k, n \in \mathbb{N}$ ,  $\text{Im}(\mathcal{D}_{k,n}(\mathcal{R})) \subseteq \mathcal{R}^n$ . Let  $\mathbf{C}$  be a probabilistic code generation algorithm such that  $\mathbf{C}(k, n, \mathcal{R})$  outputs a matrix  $H \in \mathcal{R}^{k \times n}$ . For dimension  $k = k(\lambda)$ , number of samples (or block length)  $n = n(\lambda)$ , and ring  $\mathcal{R} = \mathcal{R}(\lambda)$ , the (dual)  $(\mathcal{D}, \mathbf{C}, \mathcal{R})$ -LPN( $k, n$ ) assumption states that

$$\begin{aligned} & \{(H, \mathbf{b}) \mid H \stackrel{\$}{\leftarrow} \mathbf{C}(k, n, \mathcal{R}), \mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{D}_{k,n}(\mathcal{R}), \mathbf{b} \leftarrow H \cdot \mathbf{s}\} \\ & \stackrel{\approx}{\sim} \{(H, \mathbf{b}) \mid H \stackrel{\$}{\leftarrow} \mathbf{C}(k, n, \mathcal{R}), \mathbf{b} \stackrel{\$}{\leftarrow} \mathcal{R}^n\}. \end{aligned}$$

The dual LPN assumption is also called the *syndrome decoding assumption* in the code-based cryptography literature. The dual LPN assumption as written above is equivalent to the *primal* LPN assumption with respect to  $G$  (a matrix  $G \in \mathcal{R}^{n \times n-k}$  such that  $H \cdot G = 0$ ), which states that  $G \cdot \mathbf{s} + \mathbf{e}$  is indistinguishable from random, where  $\mathbf{s} \stackrel{\$}{\leftarrow} \mathcal{R}^{n-k}$  and  $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{D}_{k,n}(\mathcal{R})$ ; the equivalence follows from the fact that  $H(G \cdot \mathbf{s} + \mathbf{e}) = H \cdot \mathbf{e}$ .

The standard LPN assumption refers to the case where  $H$  is a uniformly random matrix over  $\mathbb{F}_2$ , and  $\mathbf{e}$  is sampled from  $\text{Ber}_r(\mathbb{F}_2)$ , where  $r$  is called the *noise rate*. Other common noise distributions include exact noise (the noise vector  $\mathbf{e}$  is a uniformly random weight- $rn$  vector from  $\mathbb{F}_2^n$ ; this is a common choice in concrete LPN-based constructions) and regular noise (the noise vector  $\mathbf{e}$  is a concatenation of  $rn$  random unit vectors from  $\mathbb{F}_2^{1/r}$ , widely used in the PCG literature [BCG18, BCG+19b, BCG+19a]).

## 2.2 The Variable Density LPN Assumption

We recall the *regular VDLPN* assumption from [BCG+20a]; other variants exist. Let  $\lambda$  be a security parameter. We fix three parameters: a *sparsity* parameter  $w = w(\lambda)$  (controlling the number of ones per row of a block), a *block* parameter  $D = D(\lambda)$  (controlling the number of blocks), and a bound  $N = N(\lambda)$  on the number of samples. The reader can think of  $w, D$  as being  $\Omega(\lambda)$ , with  $D < w$ , and  $N = 2^D$  for concreteness. We set  $\text{par} \leftarrow (w, D, N)$ .

Let  $\mathcal{S}_{1,2^i}$  the distribution of unit vector of size  $2^i$ . Let  $\mathcal{R}_{w,i}$  be the distribution of random  $w$ -regular vectors over  $\mathbb{F}_2^{w \cdot 2^i}$ , i.e., the concatenation of  $w$  vector sampled from  $\mathcal{S}_{1,2^i}$ . Let  $\mathcal{H}_{\text{par}}^i$  denote the distribution over  $N \times (w \cdot 2^i)$  matrices over  $\mathbb{F}_2$ , where each row of the matrix is sampled independently from  $\mathcal{R}_{w,i}$ , and let  $\mathcal{H}_{\text{par}}$  denote the distribution over  $\mathbb{F}_2^{N \times 2N \cdot w}$ , obtained by sampling  $H_i \stackrel{\$}{\leftarrow} \mathcal{H}_{\text{par}}^i$  for  $i = 1$  to  $D$  and outputting  $H = H_1 \parallel \dots \parallel H_D$ , where  $\parallel$  is the horizontal concatenation. Eventually we denote  $\mathcal{N}_{\text{par}}$  the noise distribution obtained by sampling  $\mathbf{e}_i^\top$  according  $\mathcal{R}_{w,i}$  and outputting  $\mathbf{e} \leftarrow (\mathbf{e}_1 \parallel \dots \parallel \mathbf{e}_D) \in \mathbb{F}_2^{2N \cdot w}$  where  $\parallel$  is this time the vertical concatenation. The matrix  $H_i$  sampled from  $\mathcal{H}_{\text{par}}^i$  is:



$$H_i = \begin{bmatrix} u_{1,1}^i & \cdots & \overbrace{u_{1,w}^i}^{2^i \text{ columns}} \\ \vdots & \vdots & \vdots \\ u_{N,1}^i & \cdots & u_{N,w}^i \end{bmatrix}$$

where  $(u_{k,j}^i)_{1 \leq k \leq N, 1 \leq j \leq w}$  are sampled from the distribution  $\mathcal{S}_{1,2^i}$ , and are unit vector over  $\mathbb{F}_2^{2^i}$ . Thus, there is  $w$  non-zero coordinates by rows. Eventually, the matrix  $H$  sampled from  $\mathcal{H}_{\text{par}}$  is a horizontal concatenation of the  $H_i$ :

$$H = \underbrace{[H_1 \cdots H_D]}_{w \cdot 2^{D+1} \text{ columns}}$$

The term *variable density* refers to the fact that the density of 1's in each block  $H_i$  is  $1/2^i$  by construction. For any  $H$  sampled from the distribution  $\mathcal{H}_{\text{par}}$  let  $\mathcal{O}_{\text{par}}(H)$  be the distribution which samples  $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{N}_{\text{par}}$  and return  $H \cdot \mathbf{e}$ .

**Definition 12** (rVDLPN( $w, D, N$ )). *The regular VDPLN assumption, with parameters  $\text{par} = (w, D, N)$ , denoted rVDLPN( $w, D, N$ ), states that:*

$$\{(H, \mathbf{b}) | H \stackrel{\$}{\leftarrow} \mathcal{H}_{\text{par}}, \mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{N}_{\text{par}}, \mathbf{b} \leftarrow H \cdot \mathbf{e}\} \approx \{(H, \mathbf{b}) | H \stackrel{\$}{\leftarrow} \mathcal{H}_{\text{par}}, \mathbf{b} \stackrel{\$}{\leftarrow} \mathbb{F}_2^N\}$$

Note that this is exactly the dual LPN assumption where both the matrix and the noise are sampled from a specific distribution variable-density matrices and vectors.

**A WPRF Candidate from the rVDLPN Assumption.** Fix parameters  $\text{par}(\lambda) = (w(\lambda), D(\lambda), N(\lambda) = 2^{D(\lambda)})$ . Recall that a vector from the distribution  $\mathcal{N}_{\text{par}}$  is in fact the vertical concatenation of  $D$  vectors from  $\mathbf{e}_i$ , where  $\mathbf{e}_i$  is the transpose vector of the vector from the distribution  $\mathcal{R}_{w,i}$ . Moreover,  $\mathcal{R}_{w,i}$  is the concatenation of  $w$  unit vector over  $\mathbb{F}_2^{2^i}$ , where each of them can be generated with  $i$  random bits (encoding the index of the nonzero entry). Therefore, sampling a vector  $\mathcal{N}_{\text{par}}$  requires exactly  $w \cdot \sum_{i=1}^D i = w \cdot D(D - 1)/2$  random bits; we write  $\mathcal{N}_{\text{par}}(r)$  to denote the vector  $\mathbf{e}$  sampled from  $\mathcal{N}_{\text{par}}$  using randomness  $r$ . We describe the WPRF candidate below.

- Key size:  $K \in \{0, 1\}^{\pi(\lambda)}$  with  $\pi(\lambda) = \rho(\lambda) = w \cdot D(D - 1)/2$
- Input size :  $x \in \{0, 1\}^{\rho(\lambda)}$  with  $\rho(\lambda) = w \cdot D(D - 1)/2$
- $F_K(x)$  : on input  $x \in \{0, 1\}^\rho$ , sample  $\mathbf{h}^\top \leftarrow \mathcal{N}_{\text{par}}(x)$  and output  $\langle \mathbf{h}, \mathcal{N}_{\text{par}}(K) \rangle$

**Theorem 13** ([BCG+20a]). *Suppose that rVDLPN( $\text{par}$ ) holds. Then the above construction is an  $N$ -query WPRF, with input length and key length equal at  $w \cdot D(D - 1)/2$ .*

### 2.3 Pseudorandom Correlation Functions

Pseudorandom correlation functions, introduced in [BCG+20a], allow to locally generate, from a pair of short correlated keys, an arbitrary polynomial amount of pseudorandom correlations, in an incremental way. A fundamental application of PCF is to secure computation in the preprocessing model: two parties can distributively generate PCF keys, and later use them every time they wish to engage in a secure computation protocol, to generate locally (without any interaction) all preprocessing material required for the protocol. Therefore, PCFs allow to confine *all* future preprocessing phases of all secure computation protocols two parties could want to execute, to a single, one time generation of short correlated keys, followed solely by local computations. Slightly more formally, a PCF is a pair  $(\text{PCF.Setup}, \text{PCF.Eval})$  where  $\text{PCF.Setup}$  generates short correlated keys  $(k_0, k_1)$ , and  $\text{PCF.Eval}(\sigma, k_\sigma, x)$  outputs a value  $y_\sigma$  such that for any input  $x$ , given  $k_{1-\sigma}$ , the value  $y_\sigma$  is indistinguishable from a random value sampled conditioned on satisfying the target correlation with  $\text{PCF.Eval}(\sigma, k_{1-\sigma}, x)$  (for  $\sigma = 0, 1$ ). Due to lack of space, we defer the definition of PCF in the full version.

### 2.4 Pseudorandom Correlation Function from VDLPN

A construction of PCF from VDLPN follows from the general template established in [BCG+20a], which combines a WPRF in a suitably low complexity class with a function secret sharing scheme for a related class. Instantiating this general template with the VDLPN-based WPRF and the FSS scheme of [BGI16], one gets a PCF for a general class of constant degree polynomial additive correlations. For the sake of concreteness, though, we focus here on PCFs for the random oblivious transfer (OT) correlation, one of the most fundamental and useful correlation in secure computation. A random OT correlation is a pair  $(y_0, y_1) \in \{0, 1\}^2 \times \{0, 1\}^2$ , where  $y_0 = (u, v)$  for two random bits  $u, v$ , and  $y_1 = (b, u \cdot (1 - b) \oplus v \cdot b)$  for a random bit  $b$ .

It is known that, to generate  $n$  pseudorandom OT correlations, it suffices to generate the following simpler correlation: Alice gets a (pseudo)random pair of length- $n$  vectors  $(\mathbf{u}, \mathbf{v})$ , where  $\mathbf{u} \xleftarrow{\$} \mathbb{F}_2^n$  and  $\mathbf{v} \in \mathbb{F}_{2^\lambda}^n$ , and Bob gets  $x \xleftarrow{\$} \mathbb{F}_{2^\lambda}$  and  $\mathbf{w} \leftarrow x \cdot \mathbf{u} + \mathbf{v}$ . This correlation (known as the *subfield vector-OLE* correlation) can be locally converted by Alice and Bob into  $n$  pseudorandom OT correlations using a correlation-robust hash function; see [BCG+19b] for details. Therefore, we focus on building a PCF for the subfield VOLE correlation. Unlike the general case, this does not require the full power of function secret sharing: it suffices to rely on a simpler primitive, namely, a puncturable pseudorandom function.

**Puncturable Pseudorandom Functions.** A *puncturable pseudorandom function* (PPRF) is a PRF  $F$  such that given an input  $x$ , and a PRF key  $k$ , one can generate a *punctured* key, denoted  $k\{x\}$ , which allows evaluating  $F$  at every point except for  $x$ , and does not reveal any information about the value  $F.\text{Eval}(k, x)$ . PPRFs have been introduced in [KPTZ13, BW13, BGI14].

Formally, a  $t$ -puncturable pseudorandom function (PPRF) with key space  $\mathcal{K}$ , domain  $\mathcal{X}$ , and range  $\mathcal{Y}$ , is a pseudorandom function  $F$  with an additional punctured key space  $\mathcal{K}_p$  and three probabilistic polynomial-time algorithms ( $F.\text{KeyGen}$ ,  $F.\text{Puncture}$ ,  $F.\text{Eval}$ ) such that

- $F.\text{KeyGen}(1^\lambda)$  outputs a random key  $K \in \mathcal{K}$ ,
- $F.\text{Puncture}(K, S)$ , on input a key  $K \in \mathcal{K}$ , and a subset  $S \subset \mathcal{X}$  of size (at most)  $t$ , outputs a punctured key  $K\{S\} \in \mathcal{K}_p$ ,
- $F.\text{Eval}(K\{S\}, x)$ , on input a key  $K\{S\}$  punctured at all points in  $S$ , and a point  $x$ , outputs  $F(K, x)$  if  $x \notin S$ , and  $\perp$  otherwise.

The (static) security of a  $t$ -PPRF states is captured by the following game: the adversary  $\mathcal{A}$  sends a size- $t$  subset  $S$  of inputs. The challenger generates a key  $K$ , a punctured key  $K\{S\}$ , and a random bit  $b$ . He sends  $K\{S\}$  to  $\mathcal{A}$ , together with either the values  $F_K(x) = F.\text{Eval}(K\{\emptyset\}, x)$  for all  $x \in S$  if  $b = 0$ , or  $t$  random bits if  $b = 1$ . The PPRF is secure if any adversary has negligible advantage over the random guess for finding  $b$  in this game. A  $t$ -PPRF can be constructed from any one-way function, using the GGM construction [GGM86].

**A PCF for SVOLE from VDLPN and a PPRF.** We briefly sketch the construction, and refer to [BCG+20a] for a formal analysis. Fix VDLPN parameters  $\text{par} = (w, D, N)$  and set  $t \leftarrow D \cdot w$ . Let  $F$  be a  $t$ -PPRF with range  $\mathbb{F}_{2^\lambda}$ .

- $\text{PCF.Setup}(1^\lambda)$  : sample  $r \xleftarrow{\$} \{0, 1\}^{t(D-1)/2}$  and set  $\mathbf{e} \leftarrow \mathcal{N}_{\text{par}}(r)$ . Let  $S \subseteq [w \cdot (2^{D+1} - 1)]$  be the size- $t$  subset of nonzero entries of  $\mathbf{e}$ . Sample  $K \leftarrow F.\text{KeyGen}(1^\lambda)$  and set  $K\{S\} \leftarrow F.\text{Puncture}(K, S)$ . Sample  $x \xleftarrow{\$} \mathbb{F}_{2^\lambda}$  and let  $(K_y)_{y \in S} \leftarrow (F_K(i) - x)_{i \in S}$ . Set  $\mathbf{k}_0 \leftarrow (K, x)$  and  $\mathbf{k}_1 \leftarrow (r, K\{S\}, (K_i)_{i \in S})$ .
- $\text{PCF.Eval}(\sigma, \mathbf{k}_\sigma, z)$  : parse  $z$  as a row  $\mathbf{h}_z$  of the VDLPN matrix  $H$  (i.e., set  $\mathbf{h}_z^\top \leftarrow \mathcal{N}(z)$ ). Let  $S_z \subseteq [w \cdot (2^{D+1} - 1)]$  denote the index of the 1's in  $\mathbf{h}_z$ . If  $\sigma = 0$ , output  $x$  and  $w = \sum_{i \in S_z} F_K(i)$ . If  $\sigma = 1$ , output  $u = \mathbf{h}_z^\top \cdot \mathbf{e}$  and  $v = \sum_{i \in S_z \setminus S} F.\text{Eval}(K\{S\}, i) + \sum_{i \in S_z \cap S} K_i$ .

For correctness, observe that for every  $i \in S_z \setminus S$ ,  $F_K(i) - F.\text{Eval}(K\{S\}, i) = 0$ , and for every  $i$  in  $S_z \cap S$ ,  $F_K(i) - K_i = x$ . Since  $S_z$  denotes the 1 entries in  $\mathbf{h}_z$  and  $S$  denotes the 1 entries in  $\mathbf{e}$ , we have  $w - v = \sum_{i \in S_z} F_K(i) - \sum_{i \in S_z \setminus S} F.\text{Eval}(K\{S\}, i) - \sum_{i \in S_z \cap S} K_i = x \cdot (\mathbf{h}_z^\top \cdot \mathbf{e}) = x \cdot u$ ; the pseudorandomness of  $u$  follows from the fact that  $z \mapsto \mathbf{h}_z^\top \cdot \mathbf{e}$  is a WPRF under the VDLPN assumption, and that of  $w$  follows from the pseudorandomness of the PPRF.

### 2.5 Outline of the Original Proof of Resistance Against Linear Test

We provide here an overview of the original security analysis in [BCG+20a], resistance against linear attacks. The two claims for which the analysis was flawed are the Eq. 1 and the Lemma 17 . We explain the errors and provide a correction in the Sect. 4.

As outlined in the introduction, the goal of this analysis is to show that the VDLPN assumption cannot be broken by any *linear test*, which captures in particular all known attacks against LPN. This is formalized in the following theorem:

**Theorem 14.** (*Resistance against linear tests*) *There exist constants  $(\Gamma, \mu, \nu)$ , such that for any large enough  $w$ , any  $\Gamma \cdot D \leq w, N \leftarrow 2^D, \text{par} \leftarrow (w, D, N)$ , it holds that*

$$\Pr_{H \leftarrow \mathcal{H}_{\text{par}}} [\text{bias}(\mathcal{O}_{\text{par}}(H)) > \mu^w] \leq \nu^w.$$

This theorem states that with high probability (at least  $1 - \nu^w$ ), over the choice of at most  $N = 2^D$  random inputs  $(x^{(1)}, \dots, x^{(N)})$  any distinguisher that computes a linear function of the entire output string  $\mathbf{y} = (F_K(x^{(1)}), \dots, F_K(x^{(N)}))$  has an advantage of at most  $\mu^w$  in distinguishing the string from uniform. Note that the choice of the linear function can depend arbitrarily on  $(x^{(1)}, \dots, x^{(N)})$ .

To bound the bias of  $\mathcal{O}_{\text{par}}(H)$ , the authors look at the sub-matrices of  $H$ , and introduce a notion of *good* and *bad* matrices:

**Definition 15.** *Given a matrix  $M \in \mathbb{F}_2^{N \times 2^i}$ ,  $M$  is judged bad with respect to a vector  $\mathbf{v} \in \mathbb{F}_2^N$  if*

$$\text{HW}(\mathbf{v}^\top \cdot M) \notin \left[ \frac{2^i}{5}, \frac{2^{i+2}}{5} \right].$$

*Moreover, given  $w$  matrices  $(M_1, \dots, M_w)$  in  $\mathbb{F}_2^{N \times 2^i}$ , we denote by  $N_{\mathbf{v}}(M_1, \dots, M_w)$  the number of matrices which are bad against  $\mathbf{v}$  among  $M_1 \dots M_w$ .*

A matrix is bad with respect to a vector  $\mathbf{v}$  if the bias it induces against the test vector  $\mathbf{v}$  is large. The goal of the proof is to guarantee that, with high probability, at least half of the matrices are good. This is stated in the following lemma.

**Lemma 16.** *There is a constant  $C$ , such that for any  $1 \leq i \leq D$ , and for any vector  $\mathbf{v} \in \mathbb{F}_2^N$  such that  $\text{HW}(\mathbf{v}) \in [2^{i-1}, 2^i]$ , it holds that*

$$\Pr_{M_1, \dots, M_w \leftarrow \mathcal{H}_{\text{par}}^i} \left[ N_{\mathbf{v}}(M_1, \dots, M_w) \geq \frac{w}{2} \right] \leq 2^{-C \cdot 2^i \cdot w}.$$

The above lemma shows that for any fixed vector  $\mathbf{v}$  of weight close to  $2^i$ , the distribution induced by  $\mathcal{H}_{\text{par}}^i$  has a low bias against  $\mathbf{v}$ . The probability that this holds is so high that it remains overwhelming even after a union bound over *all* vectors  $\mathbf{v}$  of weight in  $[2^{i-1}, 2^i]$ . Hence, this implies that in the output  $H \cdot \mathbf{e} = \bigoplus_i H_i \cdot \mathbf{e}_i$ , each component  $H_i \cdot \mathbf{e}_i$  will guarantee low-bias against all vectors in this window of weight; the XOR of these independent samples will inherit the low-bias of all its components, and therefore resist all linear tests.

**Bounding the Number of Bad Matrices.** In [BCG+20a], the authors reformulate the event that a matrix  $M$  is *bad* as a balls and bins problem. Let  $M \stackrel{\$}{\leftarrow} \mathcal{H}_{\text{par}}^i$ . Recall that by definition of  $\mathcal{H}_{\text{par}}^i$ , the rows of  $M$  are generated independently from  $\mathcal{S}_{1,2^i}$ . We start with  $2^i$  empty bins, each bin corresponding to a column of  $M$ . Sampling a row of  $M$  according to the  $\mathcal{S}_{1,2^i}$  distribution amounts to throwing a ball randomly into one of the  $2^i$  bins. For a vector  $\mathbf{v}$  of weight  $l \in [2^{i-1}, 2^i]$ , the event  $\text{HW}(\mathbf{v}^\top \cdot M) \notin \left[\frac{2^i}{5}, \frac{2^{i+2}}{5}\right] = I_i$  is equivalent to the following event: after randomly throwing  $l$  balls into  $2^i$  bins, the number  $T$  of bins that contain an odd number of balls satisfies  $T \notin I_i$ . We have therefore the following experiment: take  $2^i$  bins and throw  $l \cdot w$  balls into the bins in  $w$  consecutive phase. Each time that  $l$  balls have been thrown, we check that the proportion of the number of bins that contains an odd number of balls is between  $1/5$  and  $4/5$ , and clear out the bins. At the end, we return *failure* if more than  $w/2$  of the  $w$  checks have failed. To bound the probability of returning a failure, define the following *cost function*  $\Phi(X_{1,1}, \dots, X_{l,w}) = \sum_{k=1}^w \left(2^{i-1} - \left| \text{HW} \left( \bigoplus_{j=1}^l X_{j,k} \right) - 2^{i-1} \right| \right)$ , where each  $X_{j,k}, 1 \leq j \leq l, 1 \leq k \leq w$ , is the random variable corresponding to the bin in which the  $j$ -th balls of the  $k$ -th phase was thrown (seen as a length- $2^i$  unit vector with a 1 at the bin position). The  $X_{j,k}$  are independent. Bounding the number of bad matrices, the authors claimed, amounts to bounding  $\Phi$ . Indeed:

$$\Pr_{M_1, \dots, M_w \stackrel{\$}{\leftarrow} M_{\text{par}}^i} \left[ N_{\mathbf{v}}(M_1, \dots, M_w) \geq \frac{w}{2} \right] \leq \Pr \left[ \Phi(X_{1,1}, \dots, X_{l,w}) < \frac{w \cdot 2^i}{10} \right]. \tag{1}$$

Afterwards, it suffices to bound  $\Phi$  to conclude. The claim is that the following bound holds:

$$\Pr \left[ \Phi(X_{1,1}, \dots, X_{l,w}) < \frac{w \cdot 2^i}{10} \right] \leq 2^{-C \cdot 2^i \cdot w}. \tag{2}$$

The choice of  $\Phi$  is of course not arbitrary:  $\Phi$  is a well-behaved function, in the sense that it is 2-Lipschitz – i.e., changing any single input to  $\Phi$  can only change its output by at most 2. Fortunately, strong concentration bounds are known on the probability that Lipschitz functions deviate too much from their mean. It therefore only remains to apply such a bound (which is here the McDiarmid inequality, a variant of the Azuma inequality), to get an estimate of the mean of  $\Phi$ . This bound is stated in the following lemma:

**Lemma 17.**

$$\mathbb{E}[\Phi(X_{1,1}, \dots, X_{l,w})] \geq \frac{w \cdot 2^i}{5}.$$

Given the proof of Lemma 17, the McDiarmid inequality provides a bound on  $\Phi$ , which translates to a bound on  $N_{\mathbf{v}}$  by Eq. 1. A union bound over all vectors of weight between  $[2^{i-1}, 2^i]$  allows to conclude:

$$\Pr_{M_1, \dots, M_w \stackrel{\$}{\leftarrow} \mathcal{H}_{\text{par}}^i} \left[ \exists \mathbf{v} \in \mathcal{S}_{i,N}, N_{\mathbf{v}}(M_1, \dots, M_w) \geq \frac{w}{2} \right] \leq 2^{D \cdot 2^i} \cdot 2^{-C \cdot w \cdot 2^i} \leq 2^{-a \cdot w},$$

with  $a = \frac{C}{2} > D$ . The proof ends with a last union bound over all matrices  $H_i$ , for  $1 \leq i \leq D$ .

**Some Notations.** In the following, we will denote by  $X_{j,k}$  indicates the bin into which the  $j$ -th ball of the  $k$ -th phase is thrown ( $X_{j,k}$  is a unit vector). Given a test vector  $\mathbf{v} \in \mathbb{F}_2^N$  of weight  $\text{HW}(\mathbf{v}) = l$ , we define  $R_{i,l,k} = \text{HW}(\mathbf{v}^\top \cdot M) = \text{HW} \left( \bigoplus_{j=1}^l X_{j,k} \right)$ . That is,  $R_{i,l,k}$  it is the number of bins that contains an odd number of 1 in the  $k$ -th phase; we usually write it  $R_{l,k}$  when  $i$  is clear from the context. We further define  $Z_{i,l,k}$  as  $Z_{i,l,k} = |2^{i-1} - R_{l,k}|$  (also usually written  $Z_{l,k}$ ). Eventually, we denote by  $\mathcal{S}_{i,N}$  the set of vectors  $\mathbf{v} \in \mathbb{F}_2^N$  with  $\text{HW}(\mathbf{v}) \in [2^{i-1}, 2^i]$ .

### 3 Faster PCF from a VDLPN Variant

The original proof shows that for an appropriate choice of a constant  $\Gamma$ , if  $w \geq \Gamma \cdot D$ , then the bias of  $\mathcal{O}_{\text{par}}$  is  $2^{-\Omega(w)}$  with probability  $1 - 2^{-\Omega(w)}$ . However, the concrete constants are utterly impractical (the correction of the flaw doesn't help as we will see in Sect. 4). With a quick back-of-the-envelope calculation, to guarantee  $D \cdot 2^{-a \cdot w} < 2^{-80}$  we need  $w > \frac{85}{a}$  (for  $D = 30$ ). However, the value  $a$  in our analysis satisfies  $a < \frac{1}{40000}$ , leading to a necessary value of  $w \approx 10^6$ . These parameters are of course completely unusable. Therefore, in its current state, the proof only shows the asymptotic security of the construction in a parameter range which cannot be instantiated; any concrete instantiation is bound to rely only on heuristic parameters instead, not backed up by any security analysis. In this section, we aim at mitigating this unsatisfying situation, and provide a parameter set which is simultaneously usable in practice, and comes with provable security guarantees.

#### 3.1 A Proof Friendly VDLPN Variant for Resistance Against Linear Attack

We put forth a simple tweak of the VDLPN assumption which allows for a much tighter proof of resistance against linear attack, yet enjoys the same applications as the original VDLPN assumption. The tweak is straightforward: recall that in the original construction, the matrix  $H$  is sampled as a concatenation of matrices  $H_1 \cdots H_D$ , where each  $H_i$  is a concatenation of  $w$  matrices whose rows are unit vectors of length  $2^i$ . In the security analysis, the authors bound the bias of the  $H_i \cdot \mathbf{e}_i$  terms against length- $\Theta(2^i)$  attack vectors. However, the bounds from the new correct analysis of Sect. 4 turned out to be much worse for small constant values of  $i$  (to the point that the bounds do not suffice anymore for very small  $i$ , and we have to handle them separately). Here, we suggest replacing

$H_1 || \dots || H_{i^*-1}$ , where  $i^*$  is some fixed small constant (we will pick  $i^* = 5$  in our concrete instantiation), by a uniformly random matrix  $R$  of appropriate dimensions. That is,  $H$  is now of the form  $H = [R || H_{i^*} || \dots || H_D]$ . As before, the noise distribution will be identical to the row distribution of  $H$ . This means that we will have  $H \cdot \mathbf{e} = R \cdot \mathbf{e}_r + \sum_{i=i^*}^D H_i \cdot \mathbf{e}_i$ , where  $\mathbf{e}_r$  is a uniformly random vector.

Let  $t$  be the width of  $R$ . We show that the  $R \cdot \mathbf{e}_r$  term guarantees resistance against all low-weight tests. Then, saying that the distribution  $\mathcal{D}_R = \{R \cdot \mathbf{e}_r : \mathbf{e}_r \xleftarrow{\$} \mathbb{F}_2^t\}$  has zero bias against all vectors of weight below  $d$  is equivalent to saying that  $\mathcal{D}_R$  is a  $d$ -wise independent distribution. It is a well-known fact that this is equivalent to the following: the dual of the code generated by  $R$ , which is a random linear code of dimension  $2^D - t$ , has minimum distance at least  $d$ . Fortunately, the minimum distance of random linear codes is well-known. Let  $S$  be a random code of dimension  $2^D - t$ , and codeword length  $2^D$ . Then,

$$\Pr[S \text{ has minimum distance } < d] \leq 2^{-t - H_2(d/2^D) \cdot 2^D},$$

where  $H_2(x) = -x \log x - (1 - x) \log(1 - x)$  is the binary entropy function. Concretely, suppose that we want to perfectly withstand all linear tests of weight at most  $d = 15$ , with probability at least  $1 - 2^{-\lambda}$ , given up to  $2^D = 2^{30}$  samples. This means we need to pick  $t$  such that  $t = H_2(15/2^{30}) * 2^{30} + \lambda$ ; using  $\lambda = 128$ , this gives  $t = 541$ . Hence, picking a uniformly random width-541 matrix guarantees that, with probability at least  $1 - 2^{-128}$ , we only have to worry about any linear test of weight at least  $16 = 2^{i^* - 1}$ . Note that this variant can be used exactly in the same way as the original VDPLN one, as a building block to construct PCF, as long as we can prove its security against linear tests.

### 3.2 A New Tight Proof Strategy

For the rest of the analysis, we assume that we start with  $i \geq i^* = 5$ . The adversary chooses an attack vector  $\mathbf{v}$  of hamming weight  $l \in [2^{i-1}, 2^i]$ . We use the following random variable:

$$Z_{i,l,k} = \left| \text{HW} \left( \bigoplus_{j=1}^l X_{j,k}^{(i)} \right) - 2^{i-1} \right|.$$

Unlike the original proof (see Sect. 2.5), this time we aim at a much more direct strategy. Since we ultimately want to bound the probability that the bias of  $\mathcal{O}_{\text{par}}(H)$  is too high, we rewrite this bias directly in terms of the above random variable. For a fixed choice of  $H$ , let  $\mathcal{O}_{\text{par}}^i = \mathcal{O}_{\text{par}}^i(H)$  be the distribution that samples  $\mathbf{e}_i$  (a concatenation of  $w$  length- $2^i$  unit vectors) and outputs  $H_i \cdot \mathbf{e}_i$ . Of course, we have  $\mathcal{O}_{\text{par}} = \bigoplus_{i \geq i^*} \mathcal{O}_{\text{par}}^i \oplus \mathcal{O}_{\text{par}}^R$  (where  $\mathcal{O}_{\text{par}}^R$  denotes the distribution that samples a uniformly random length- $t$  vector  $\mathbf{e}_r$  and outputs  $R \cdot \mathbf{e}_r$ , where  $t$  is a parameter which will be fixed afterwards). Furthermore, for any test vector

$\mathbf{v}$ , we have  $\text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}) \geq \text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i)$ . We therefore focus on bounding the bias against a test vector  $\mathbf{v}$  of  $\mathcal{O}_{\text{par}}^i$ . We have

$$\text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) = \left| \frac{1}{2} - \Pr[(\mathbf{v}^\top \cdot H_i) \cdot \mathbf{e}_i = 1] \right|.$$

$(\mathbf{v}^\top \cdot H_i) \cdot \mathbf{e}_i$  is the XOR of  $w$  independent terms  $(\mathbf{v}^\top \cdot H_{i,j}) \cdot \mathbf{e}_{i,j}$  where each  $\mathbf{e}_{i,j}$  is a length- $2^i$  unit vector. Therefore, we further decompose  $\mathcal{O}_{\text{par}}^i$  as the XOR of  $w$  distributions  $D_1, \dots, D_w$  (we drop the parameters  $i, \text{par}$ , and  $H$  for now as we consider a fixed choice of them, to lighten the notations). To bound the bias of  $\mathcal{O}_{\text{par}}^i$ , we must therefore bound

$$\Pr \left[ \bigoplus_{k=1}^w D_k = 1 \right] = \frac{1}{2} \left( 1 - \prod_{k=1}^w \left( 1 - \frac{R_{i,l,k}}{2^{i-1}} \right) \right),$$

where you get the right hand side by applying the piling-up lemma. Hence, we obtain a direct expression of the bias of  $\mathcal{O}_{\text{par}}^i$  in terms of the  $Z_{i,l,k}$  random variables:

$$\text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) = \frac{1}{2} \cdot \prod_{k=1}^w \frac{Z_{i,l,k}}{2^{i-1}}.$$

Fix any bound  $B$ . Then by the above,

$$\Pr[\text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) > B] = \Pr \left[ \prod_{k=1}^w Z_{i,l,k} > 2^{(i-1)w} \times (2B) \right].$$

Now, the key to bounding the right hand side term is the following observation: independently of the exact behavior of the random variables  $Z_{i,l,k}$ , constrained on the product  $\prod_k Z_{i,l,k}$  being at least  $2^{(i-1)w} \cdot (2B)$ , the sum  $\sum_k Z_{i,l,k}$  is minimized when all the terms in the product are equal. This implies that whenever  $\prod_{k=1}^w Z_{i,l,k} > 2^{(i-1)w} \times (2B)$ , it necessarily further holds that

$$\sum_{k=1}^w Z_{i,l,k} > w \cdot \left( 2^{(i-1)w} \times (2B) \right)^{1/w},$$

which allows to upper bound the probability of the bias being too large by

$$\Pr[\text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) > B] \leq \Pr \left[ \sum_{k=1}^w Z_{i,l,k} > w \cdot 2^{(i-1)} \cdot c \right],$$

where  $c = (2B)^{\frac{1}{w}}$ . As in the previous proof, we can now re-introduce the function  $\Phi(X_{1,1}, \dots, X_{l,w}) = 2^{i-1} \cdot w - \sum_{k=1}^w Z_{i,l,k}$ :

$$\Pr[\Phi < \mathbb{E}[\phi] - t] = \Pr \left[ \sum_{k=1}^w Z_{i,l,k} > w \cdot (\mathbb{E}[Z_{i,l}] + 2^i \cdot \zeta) \right].$$



With  $t = \zeta \cdot w \cdot 2^i$ . Let  $\beta$  be a constant such that  $\mathbb{E}[Z_{i,l}] \leq \beta \cdot 2^i$  (In the correction of the original proof, we will show that  $\beta = 0.44$  works ; we will actually use a tighter constant here). This gives  $c = 2(\beta + \zeta)$ . As we did before, we can now apply McDiarmid’s inequality 7 to get

$$\Pr \left[ \sum_{k=1}^w Z_{i,l,k} > w \cdot 2^{i-1} \cdot c \right] < \exp \left( -w \frac{2^{2i-1}}{l} \cdot \zeta^2 \right),$$

and obtain the bound

$$\Pr \left( \text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) > \frac{1}{2}(2(\beta + \zeta))^w \right) \leq \exp \left( -w \frac{2^{2i-1}}{l} \cdot \zeta^2 \right).$$

While this might be obscured by the many variables involved, this last bound is tremendously tighter than what was achieved with the previous proof. In essence, this is because the previous proof relied on Lemma 5 to bound the bias of the XOR of independent distributions, but the latter introduces some exponential slackness in the *number* of distributions involved. To overcome this slackness, the strategy was to only “count” the distributions that contribute the most to the bias, by identifying *good* distributions, showing that, over the choice of  $H$ , a sufficient number of distributions will be good, and applying the lemma only to these good distributions. This guarantees that the slackness is compensated by the contribution of each distribution. If, instead, one tries to apply the lemmas to all distribution, the bound obtain is too loose and does not provide any usable guarantee.

Here, we manage to directly account for the contribution to the bias of *all* distributions, by carefully rewriting the bias formula in terms of the  $Z_{i,l,k}$  random variables, and by using a standard “optimization trick” to bound the product of the  $Z_{i,l,k}$  in terms of their sum. This turns out to be the key to get back to the function which we can bound with known tools (the function  $\Phi$ , which is Lipschitz), without paying any slackness in the number of distributions involved.

In the following, we will numerically evaluate the constant  $\beta$  (this is an orthogonal optimization: In the correction of the mistake in Sect. 4, we prove that  $\beta \leq 0.44$ . Using this value for  $\beta$  would already lead to significant improvements, as we will see) and carefully tune the parameters to find out the smallest value of  $w$  for which we can achieve 80-bit security against all test vectors simultaneously, fixing the number of samples to a reasonable bound of  $N = 2^{30}$ .

### 3.3 Concrete Parameters

In our bound on the bias, the  $l$  in the denominator of the probability is one of the key factors for concrete efficiency. Our previous proof used  $l \in [2^{i-1}, 2^i]$ . In fact, we cannot expect  $l$  to be any smaller:  $\mathbb{E}[Z_{i,l}]$  measures how, when one throws  $l$  balls at random in  $2^i$  bins, the number of bins which end up containing an odd number of balls diverges from the middle value  $2^{i-1}$ . When we throw less than  $2^{i-1}$  balls in total, this number will of course be bounded away from  $2^{i-1}$ ; yet,

as simulations reveal, it becomes tightly concentrated around  $2^{i-1}$  as soon as  $l$  gets larger. We therefore fix  $l \in [2^{i-1}, 2^i]$  and empirically estimate  $\mathbb{E}[Z_{i,l}]$ . Our script can be found in the full version. Table 1 shows the value of  $\beta$  obtained for different choices of  $n = 2^i$  and  $l$ . For larger values of  $n$  and a fixed  $l = l(n)$ , note that our estimate value for  $\beta$  barely increase (for  $l < n$ ) or decrease (for  $l \geq n$ ).

**Table 1.** Estimated value of  $\beta$  for different values of  $n$  and  $l$ , in a confidence interval of 99% (rounded value  $\pm 0.002$ )

	$n = 32$	$n = 64$	$n = 512$	$n = 1024$	$n = 2048$
$l = \frac{n}{2}$	0.178	0.181	0.184	0.184	0.184
$l = \frac{3 \cdot n}{4}$	0.111	0.111	0.111	0.111	0.112
$l = n$	0.084	0.073	0.067	0.067	0.067

Let us go back to our bound. For a given vector of Hamming weight  $l$ ,

$$\Pr \left( \text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) > \frac{1}{2}(2(\beta + \zeta))^w \right) \leq \exp \left( -w \frac{2^{2i-1}}{l} \cdot \zeta^2 \right),$$

hence, by a union bound over all vectors of Hamming weight  $l$ ,

$$\Pr \left( \exists \mathbf{v}, \text{HW}(\mathbf{v}) = l, \text{bias}_{\mathbf{v}}(\mathcal{O}_{\text{par}}^i) > \frac{1}{2}(2(\beta + \zeta))^w \right) \leq \binom{N}{l} \exp \left( -w \frac{2^{2i-1}}{l} \cdot \zeta^2 \right)$$

From the above inequality, we numerically look for a  $w$  such that, for all  $l$  such that  $\text{HW}(l) \in [2^{i-1}, 2^i]$ ,  $(2 \cdot (\beta + \zeta))^w / 2 \leq 2^{-80}$  and

$$\binom{N}{l} \cdot \exp \left( -w \frac{2^{2i-1}}{l} \cdot \zeta^2 \right) \leq \exp \left( \ln(N) \cdot l - w \cdot \frac{2^{2i-1}}{l} \cdot \zeta^2 \right) \leq 2^{-90}.$$

(The  $2^{-90}$  bound is to anticipate the cost of the union bound.) In the following we set the number of samples  $N = 2^D = 2^{30}$ , which is a realistic value for target applications. To find a suitable  $w$ , we calculate the required  $w$  for different values of  $l$ . Let us first assume that  $l = 2^{i-1}$ . The second inequality can be rewritten as  $\exp(2^{i-1} \cdot (\ln(2) \cdot 30 - 2 \cdot w \cdot \zeta^2)) \leq 2^{-90}$ . If  $w \cdot \zeta^2 \geq 12.35$ , then the condition is met. Thus, we can now turn to the other inequality to satisfy; we therefore set  $w$  to  $\zeta^2 / 12.35$ . Using Table 1, we set  $\beta = 0.184$  and numerically solve  $(2 \cdot (0.184 + \zeta))^{12.35/\zeta^2} \leq 2^{-80}$  to guarantee that the bias will be lower than  $2^{-80}$ . This gives  $\zeta \leq 0.219$  and  $w = 12.35 / 0.219^2 \approx 257$ . At the other end of the interval, setting  $l = 2^i$ , the second inequality becomes  $\exp(2^i \cdot (\ln(2) \cdot 30 - \frac{1}{2} w \cdot \zeta^2)) \leq 2^{-90}$ .

This time, we get  $w \cdot \zeta^2 \geq 45.5$  and set  $\beta = 0.084$  using Table 1. Solving  $(2 \cdot (0.084 + \zeta))^{\frac{45.5}{\zeta^2}} \leq 2^{-80}$  gives  $\zeta = 0.347$ , and eventually  $w = 45.5 / 0.347^2 \approx 380$ . Generalizing this method, we numerically extrapolate how the value of  $w$  evolves when  $l$  varies from  $2^{i-1}$  to  $2^i$ . The calculations show that  $w$  is monotonously

increasing, leading to an overall choice of  $w \approx 380$  as a single parameter that suffices for the entire range of values. This is a major improvement compared to the previous proof.<sup>4</sup> From here, finishing the proof boils down to two union bounds, giving

$$\begin{aligned} & \Pr(\exists \mathbf{v}, \text{HW}(\mathbf{v}) \in [2^{i-1}, 2^i], \text{bias}_{\mathbf{v}} > 2^{-80}) \\ & \leq \sum_{l=2^{i-1}}^{2^i} \binom{N}{l} \exp\left(-w \frac{2^{2i-1}}{l} \cdot \zeta^2\right) \leq 2^{i^*-1} \cdot 2^{-90} = 2^{-86}, \end{aligned}$$

where the last inequality comes from the fact that  $2^{i^*} = 2^5$ . For  $i > i^*$ , the bound in the probability decreases (exponentially) faster than the increase of  $2^i$ , and the result remains valid. Eventually, by a union bound on all  $i$ , with  $D = 30$ ,  $\Pr(\exists \mathbf{v}, \text{HW}(\mathbf{v}) \geq 2^4, \text{bias}_{\mathbf{v}} > 2^{-80}) \leq (D - 8) \cdot 2^{-86} < 2^{-80}$ . Concrete cost estimations for the pseudorandom correlation function obtained by using the VDLPN parameters of our improved analysis are given in the introduction.

### 4 Security of VDLPN Against Linear Tests, Revisited

As pointed out, the analysis of [BCG+20a], while the proof strategy seems sound and appropriate, contains some errors which invalidate the proof. Fixing the errors turns out to be quite delicate. Below, we elaborate on the two issues; the first is a minor error, which can be fixed relatively easily, at the cost of changing the (arbitrary) choice of constants (in particular, the 1/5 and 1/10 constants): Claim 1 is incorrect as stated; the error in its analysis stems from a reversed inequality. However, a variant of Claim 1 with different constants can be easily shown to hold; this does not change the spirit of the proof, nor its conclusion. The second error is more delicate to fix, and will be the main focus of this Section.

**Main Error.** The main error appears in the proof of Eq. 2. The error is in the analysis of Lemma 17. As sketched in the introduction, after calculating an upper bound on the expectation  $\mathbb{E}\left[\text{HW}\left(\bigoplus_{j=1}^l X_{j,k}\right)\right]$ , the authors deduce a bound on  $\mathbb{E}\left[\left|2^{i-1} - \text{HW}\left(\bigoplus_{j=1}^l X_{j,k}\right)\right|\right]$ . However, a bound on  $\mathbb{E}[Z]$  does not imply a bound on  $\mathbb{E}[|Z - b|]$  in general (and typically when  $Z$  is “concentrated away” from  $b$ ). Up to the choice of the constant 1/5 (the proof actually only requires any constant below 1/2), the lemma remains true; however, proving the lemma fundamentally requires characterizing the shape of the random variable  $\text{HW}\left(\bigoplus_{j=1}^l X_{j,k}\right)$ . This turns out to be non-trivial.

<sup>4</sup> The improvement comes from a better estimation of the  $\beta$  parameter on one hand, but also from the better inherent quality of the new proof. In fact, we can consider the same calculation as before, but with  $l = 2^7$  and  $\beta = 0.44$ . This is a non-computer-optimized, provable value of  $\beta$  using  $i^* = 7$ . With this value, we get  $w \approx 13000$ , which is already a big gain from the previous method: this is already several orders of magnitudes better than the previous method, though still not practical.

### 4.1 Repairing the Proof

In this section, we put forward a corrected detailed analysis of the resistance of VDLPN against linear tests. Our proof fixes the two errors in [BCG+20a], at the cost of achieving worse constants, and being more involved. As before, we study individually the bias induced by the  $H_i$  components against vectors of weight close to  $2^i$ . However, for now, we only consider large enough values of  $i$ , and assume that  $n = 2^i \geq 2^7$ . The missing cases are handled in the full version.

**Definition 18 ( $\delta$ -Bad Matrices).**

Let  $M \in \mathbb{F}_2^{N \times 2^i}$ . We say that  $M \in \text{Bad}_{\delta, \mathbf{v}}$  with respect to a vector  $\mathbf{v} \in \mathbb{F}_2^N$  if

$$\text{HW}(\mathbf{v}^\top \cdot M) = R_{l,k} \notin [\delta \cdot 2^i, (1 - \delta) \cdot 2^i].$$

Stated in terms of  $Z_{l,k}$ , this condition rewrites to  $Z_{l,k} \in [(1/2 - \delta) \cdot 2^i, 2^{i-1}]$ . We let  $\text{Good}_{\delta, \mathbf{v}}$  denote the complement of  $\text{Bad}_{\delta, \mathbf{v}}$ . Given vector  $\mathbf{v}$ , we also denote  $B_{\delta, \mathbf{v}} = \#\text{Bad}_{\delta, \mathbf{v}} = N_{\mathbf{v}}(M_1, \dots, M_w)$  and  $G_{\delta, \mathbf{v}} = \#\text{Good}_{\delta, \mathbf{v}} = w - B_{\delta, \mathbf{v}}$ .

**The Proof.** We now prove Theorem 14. Let  $\mathcal{O}_{\text{par}}^i(H)$  be the distribution induced by sampling  $\mathbf{e}_i$  (as a concatenation of  $w$  length- $2^i$  vectors) and outputting  $H_i \cdot \mathbf{e}_i$ . A sample from  $\mathcal{O}_{\text{par}}^i(H)$  can be further decomposed as  $\bigoplus_{j \leq w} H_{i,j} \cdot \mathbf{e}_{i,j}$  where the  $\mathbf{e}_{i,j}$  are unit vectors. Let  $D_i$  denote the distribution of  $H_{i,j} \cdot \mathbf{e}_{i,j}$  (these terms are  $w$  samples from the same distribution). Let  $\alpha$  be a constant. Then,

**Lemma 19.** *If  $B_{\delta, \mathbf{v}} \leq \alpha \cdot w$ , then*

$$\text{bias} \left( \bigoplus_{i=1}^w D_i \right) \leq \frac{1}{2} \cdot ((1 - 2\delta)^{(1-\alpha)w}).$$

*Proof.* By the piling-up lemma (Lemma 6),

$$\text{bias} \left( \bigoplus_{i=1}^w D_i \right) \leq 2^{(1-\alpha)w-1} \cdot \left( \frac{1}{2} - \delta \right)^{(1-\alpha)w} \leq \frac{1}{2} \cdot ((1 - 2\delta)^{(1-\alpha)w})$$

□

Lemma 19 provides an upper bound of the bias, which depends on the number of good matrices and their quality. We now show that the condition  $B_{\delta, \mathbf{v}} \leq \alpha \cdot w$  holds with very high probability:

**Lemma 20.** *For any  $\mathbf{v} \in S_{i,N}$ , there is a constant  $C$  such that*

$$\Pr \left[ B_{\delta, \mathbf{v}} > \alpha \cdot w \right] \leq 2^{-C \cdot 2^i \cdot w}.$$

*Proof.* As in the original proof, we introduce the function  $\Phi$ :

$$\begin{aligned} \Phi(X_{1,1}, \dots, X_{l,w}) &= \sum_{k=1}^w \left( 2^{i-1} - \left| \text{HW} \left( \bigoplus_{j=1}^l X_{j,k} \right) - 2^{i-1} \right| \right) \\ &= 2^{i-1} \cdot w - \sum_{k=1}^w Z_{l,k}. \end{aligned}$$

We want to bound the probability of large bias by a bound on  $\Phi$ . This is where the first error appeared in the previous proof.

**Lemma 21 (Correction of the first error).**

$$\Pr \left[ B_{\delta, \mathbf{v}} \geq \alpha \cdot w \right] \leq \Pr \left[ \Phi(X_{1,1}, \dots, X_{l,w}) < \gamma \cdot w \cdot 2^i \right],$$

with  $\gamma = \frac{1}{2} - \alpha(\frac{1}{2} - \delta)$ .

Due to lack of space, the proof of the above lemma will appear in the full version. It remains now to find an upper bound on the right hand side probability. As in the original proof, we used the bounded difference inequality. Since  $\Phi$  is 2-Lipschitz, (this was proved in the original proof),

$$\Pr[\Phi(X_{1,1}, \dots, X_{l,w}) \leq \mathbb{E}[\Phi(X_{1,1}, \dots, X_{l,w})] - t] \leq \exp \left( - \frac{t^2}{2lw} \right).$$

We finally want to prove a lower bound on  $\mathbb{E}[\phi(X_{1,1}, \dots, X_{l,w})]$ . Recall that  $\Phi(X_{1,1}, \dots, X_{l,w}) = 2^{i-1} \cdot w - \sum_{k=1}^w Z_{l,k}$ , so this reduces to bounding  $\mathbb{E}[Z_{l,k}]$ . Our main contribution in this analysis is the proof of the following lemma:

**Lemma 22 (Correction of the second error).** *For all  $n \in \mathbb{N}$ , there exists  $\beta < 1/2$  such that  $\mathbb{E}[Z_{l,k}] < \beta \cdot n$ .*

*Proof (Sketch).* We first provide a high level overview, and due to lack of space, we defer the full proof of Lemma 22 to the full version of the article. The proof consists in finding an upper bound on both  $\Pr[R_{l,k} \geq p \cdot n]$  and  $\Pr[R_{l,k} \leq (1 - p) \cdot n]$  for  $p \in [\frac{1}{2}, 1]$  and to use it to find the one on  $\mathbb{E}[Z_{l,k}] = \sum_{j=0}^{2^{i-1}-1} \Pr(|R_{l,k} - 2^{i-1}| > j)$ .

**Lemma 23.** *Let  $n = 2^i > 2^7$ ,  $l \in [2^{i-1}, 2^i]$  and  $\mu = (1 - \frac{1}{n})^l$ . There exists  $0.5 \leq p \leq 1$  such that with  $\theta = \frac{pn-l/2}{\mu} - 1$ , it holds that*

$$\max(\Pr[R_{l,k} \geq pn], \Pr[R_{l,k} \leq (1 - p)n]) \leq 2 \exp \left( - \frac{\theta^2 \mu^2 (n - \frac{1}{2})}{n^2 - \mu^2} \right).$$

To prove this lemma, we use the Occupancy Bound for balls and bins from Lemma 8. The occupancy bound is about the proportion of empty bins, but can shrewdly be transformed to bring it back to our specific problem which focuses on parity in bins. This concludes the sketch. □

The end of the proof is the same as in the original proof, up to handling separately the case of small  $i$ 's. The total number of vectors  $\mathbf{v} \in S_{i,N}$  can be bounded by

$$\sum_{l=2^{i-1}}^{2^i} \binom{N}{l} \leq (2^i - 2^{i-1}) \cdot \frac{N^{2^i}}{(2^{i-1})!} \leq 2^{D \cdot 2^i}.$$

Hence, choosing constant such that  $Cw/2 > D$ , and setting  $a = C/2$ , by a union bound, we have

$$\Pr \left[ \exists \mathbf{v} \in S_{i,N}, B_{\delta, \mathbf{v}} \geq \alpha \cdot w \right] \leq 2^{D \cdot 2^i} \cdot 2^{-C \cdot 2^i \cdot w} \leq 2^{-a \cdot w}.$$

We eventually use a union bound again on all values of  $i \leq D$ :

$$\Pr \left[ \exists i \leq D, \mathbf{v} \in S_{i,N}, B_{\delta, \mathbf{v}} \geq \alpha \cdot w \right] \leq D \cdot 2^{-a \cdot w}.$$

which, using Lemma 19, rewrites to

$$\Pr \left[ \exists i \leq D, \mathbf{v} \in S_{i,N}, \text{bias}_{\mathbf{v}} \left( \bigoplus_{i=1}^w D_i \right) \geq \frac{1}{2} \cdot ((1 - 2\delta)^{(1-\alpha)})^w \right] \leq D \cdot 2^{-a \cdot w}.$$

The argument for small values of  $i$  is completely different. In essence, we show that the first block of  $H$ ,  $H_1$ , does already suffice to withstand all even-weight test vectors. Then, with a “brute-force” union bound, we show that the second block  $H_2$  allows to withstand all tests of *odd* weight, provided that  $w = \Omega(2^i \cdot D)$ . When  $i$  is a constant, this is already captured by the requirement that  $w \geq \Gamma \cdot D$  for a suitable constant  $\Gamma$ , which suffices to handle all remaining corner cases. Refer to the full version for complete explanation  $\square$

**Acknowledgements.** The first author acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE), and of the France 2030 ANR Project ANR-22-PECY-003 SecureCompute. The second author was supported by the DIM RFSI grant LICENCED.

## References

- [AJ01] Al Jabri, A.A.: A statistical decoding algorithm for general linear block codes (2001)
- [Azu67] Azuma, K.: Weighted sums of certain dependent random variables. *Tohoku Math. J. Second Series* **19**(3), 357–367 (1967)
- [BCG+17] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Orrù, M.: Homomorphic secret sharing: Optimizations and applications. In: Thuringham, B.M., Evans, D., Malkin, T., Xu, D., (eds.) *ACM CCS 2017*, pp. 2105–2122. ACM Press (October/November 2017)
- [BCG+19a] Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J., (eds.) *ACM CCS 2019*, pp. 291–308. ACM Press (November 2019)
- [BCG+19b] Boyle, E., et al.: Efficient pseudorandom 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](https://doi.org/10.1007/978-3-030-26954-8_16)

- [BCG+20a] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Correlated pseudorandom functions from variable-density LPN. In: 61st FOCS, pages 1069–1080. IEEE Computer Society Press (November 2020)
- [BCG+20b] 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](https://doi.org/10.1007/978-3-030-56880-1_14)
- [BCGI18] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X., (eds.) ACM CCS 2018, pp. 896–912. ACM Press (October 2018)
- [BGI14] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54631-0\\_29](https://doi.org/10.1007/978-3-642-54631-0_29)
- [BGI15] 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](https://doi.org/10.1007/978-3-662-46803-6_12)
- [BGI16] Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S., (eds.) ACM CCS 2016, pp. 1292–1303. ACM Press (October 2016)
- [BJMM12] Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in  $2^{n/20}$ : How  $1 + 0$  improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_31](https://doi.org/10.1007/978-3-642-29011-4_31)
- [BKW00] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: 32nd ACM STOC, pp. 435–440. ACM Press (May 2000)
- [BLP11] Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 743–760. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22792-9\\_42](https://doi.org/10.1007/978-3-642-22792-9_42)
- [BM97] Bellare, M., Micciancio, D.: A new paradigm for collision-free hashing: incrementality at reduced cost. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 163–192. Springer, Heidelberg (1997). [https://doi.org/10.1007/3-540-69053-0\\_13](https://doi.org/10.1007/3-540-69053-0_13)
- [BM18] Both, L., May, A.: Decoding linear codes with high error rate and its impact for LPN security. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 25–46. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-79063-3\\_2](https://doi.org/10.1007/978-3-319-79063-3_2)
- [BTV16] Bogos, S., Tramer, F., Vaudenay, S.: On solving lpn using bkw and variants (2016)
- [BV16] Bogos, S., Vaudenay, S.: Optimization of LPN solving algorithms. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 703–728. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_26](https://doi.org/10.1007/978-3-662-53887-6_26)

- [BW13] Boneh, D., Waters, B.: constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-42045-0\\_15](https://doi.org/10.1007/978-3-642-42045-0_15)
- [CRR21] Couteau, G., Rindal, P., Raghuraman, S.: Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC Codes. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12827, pp. 502–534. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84252-9\\_17](https://doi.org/10.1007/978-3-030-84252-9_17)
- [DAT17] Debris-Alazard, T., Tillich, J.-P.: Statistical decoding (2017)
- [DHRW16] Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 93–122. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53015-3\\_4](https://doi.org/10.1007/978-3-662-53015-3_4)
- [DPSZ12] Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32009-5\\_38](https://doi.org/10.1007/978-3-642-32009-5_38)
- [EKM17] Esser, A., Kübler, R., May, A.: LPN decoded. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 486–514. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63715-0\\_17](https://doi.org/10.1007/978-3-319-63715-0_17)
- [FKI06] Fossorier, M.P.C., Kobara, K., Imai, H.: Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem (2006)
- [FS09] Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10366-7\\_6](https://doi.org/10.1007/978-3-642-10366-7_6)
- [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](https://doi.org/10.1007/978-3-642-55220-5_35)
- [GJL20] Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. *J. Cryptol.* **33**(1), 1–33 (2020)
- [Kir11] Kirchner, P.: Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377 (2011). <https://eprint.iacr.org/2011/377>
- [KMPS94] Kamath, A., Motwani, R., Palem, K.V., Spirakis, P.G.: Tail bounds for occupancy and the satisfiability threshold conjecture. In: 35th FOCS, pp. 592–603. IEEE Computer Society Press (November 1994)
- [KPTZ13] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.-R., Gligor, V.D., Yung, M., (eds.) ACM CCS 2013, pp. 669–684. ACM Press (November 2013)
- [LF06] Lévêillé, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006). [https://doi.org/10.1007/11832072\\_24](https://doi.org/10.1007/11832072_24)
- [Lyu05] Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem (2005)



- [McD89] McDiarmid, C.: On the method of bounded differences. In: Simons, J., (ed.) "Survey in Combinatorics," London Mathematical Society Lecture Notes, vol. 141 (1989)
- [MMT11] May, A., Meurer, A., Thomae, E.: Decoding random linear codes in  $\tilde{\mathcal{O}}(2^{0.054n})$ . In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107–124. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-25385-0\\_6](https://doi.org/10.1007/978-3-642-25385-0_6)
- [MO15] May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 203–228. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_9](https://doi.org/10.1007/978-3-662-46800-5_9)
- [MSY21] Münch, J.-P., Schneider, T., Yalame, H.: Vasa: Vector aes instructions for security applications. In: Annual Computer Security Applications Conference, pp. 131–145 (2021)
- [OSY21] Orlandi, C., Scholl, P., Yakubov, 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](https://doi.org/10.1007/978-3-030-77870-5_24)
- [Ove06] Overbeck, R.: Statistical decoding revisited. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 283–294. Springer, Heidelberg (2006). [https://doi.org/10.1007/11780656\\_24](https://doi.org/10.1007/11780656_24)
- [Pra62] Prange, E.: The use of information sets in decoding cyclic codes (1962)
- [Saa07] Saarinen, M.-J.O.: Linearization attacks against syndrome based hashes. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 1–9. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-77026-8\\_1](https://doi.org/10.1007/978-3-540-77026-8_1)
- [SGRR19] Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J., (eds.) ACM CCS 2019, pp. 1055–1072. ACM Press (November 2019)
- [Shp09] Shpilka, A.: Constructions of low-degree and error-correcting  $\varepsilon$ -biased generators (2009)
- [Ste88] Stern, J.: A method for finding codewords of small weight (1988)
- [Wag02] Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
- [Zic17] Zichron, L.: Locally computable arithmetic pseudorandom generators (2017)
- [ZJW16] Zhang, B., Jiao, L., Wang, M.: Faster algorithms for solving LPN. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. Part I, volume 9665 of LNCS, pp. 168–195. Springer, Heidelberg (2016)



# Threshold Private Set Intersection with Better Communication Complexity

Satrajit Ghosh<sup>1</sup> and Mark Simkin<sup>2(✉)</sup>

<sup>1</sup> Indian Institute of Technology Kharagpur, Kharagpur, India

<sup>2</sup> Ethereum Foundation, Aarhus, Denmark

`mark.simkin@ethereum.org`

**Abstract.** Given  $\ell$  parties with sets  $X_1, \dots, X_\ell$  of size  $n$ , we would like to securely compute the intersection  $\cap_{i=1}^{\ell} X_i$ , if it is larger than  $n - t$  for some threshold  $t$ , without revealing any other additional information. It has previously been shown (Ghosh and Simkin, Crypto 2019) that this function can be securely computed with a communication complexity that only depends on  $t$  and in particular does not depend on  $n$ . For small values of  $t$ , this results in protocols that have a communication complexity that is sublinear in the size of the inputs. Current protocols either rely on fully homomorphic encryption or have an at least quadratic dependency on the parameter  $t$ .

In this work, we construct protocols with a quasilinear dependency on  $t$  from simple assumptions like additively homomorphic encryption and oblivious transfer. All existing approaches, including ours, rely on protocols for computing a single bit, which indicates whether the intersection is larger than  $n - t$  without actually computing it. Our key technical contribution, which may be of independent interest, takes any such protocol with secret shared outputs and communication complexity  $\mathcal{O}(\lambda \ell \text{poly}(t))$ , where  $\lambda$  is the security parameter, and transforms it into a protocol with communication complexity  $\mathcal{O}(\lambda^2 \ell t \text{polylog}(t))$ .

## 1 Introduction

In the private set intersection (PSI) setting,  $\ell$  parties with private input sets  $X_1, \dots, X_\ell$  would like to jointly compute  $\cap_{i=1}^{\ell} X_i$  without revealing anything else about any of the sets to each other. PSI is a powerful tool with applications in various places, such as botnet detection [NMH+10], online advertising [PSSZ15], private contact discovery [Mar14], and contact tracing [DPT20]. Various works have shown how to design asymptotically and practically efficient protocols in both the two and multiparty setting with security against both passive and active adversaries [Mea86, FNP04, KS05, DCW13, PSSZ15, KKRT16, PRTY19, PRTY20]. Unfortunately, all these protocols have communication complexities that are at least linear in the size of the smallest input set and it was observed by Freedman, Nissim, and Pinkas [FNP04] that one cannot hope to do better in general.

Ghosh and Simkin [GS19] have recently shown that the communication complexity can be sublinear in the sizes of the input sets, when the intersection is

very large. The authors considered the threshold private set intersection (TPSI) setting, where the parties would like to compute the intersection of their sets, if and only if it is larger than  $n - t$ , where  $n$  is the size of each set and  $t$  is some threshold. Based on simple assumptions, such as the existence of oblivious transfer and additively homomorphic encryption, Ghosh and Simkin construct protocols for TPSI with a communication complexity of  $\mathcal{O}(\lambda t^2 \text{polylog } t)$  bits, where  $\lambda$  is the computational security parameter, for the two-party case. The authors also show how to construct a close to optimal two-party protocol based on fully homomorphic encryption with  $\mathcal{O}(\lambda t \text{polylog } t)$  bits and outline how these protocols can be extended to the multiparty case. The authors show an  $\Omega(t)$  lower bound on the communication complexity for the two-party case. Subsequently Branco, Döttling, and Pu [BDP21] present an  $\ell$ -party protocol with a communication complexity of  $\mathcal{O}(\lambda \ell t^2 \text{polylog } t)$  bits based on threshold additively homomorphic encryption. Badrinarayanan et al. [BMRR21] propose a protocol for a setting similar to the TPSI setting above, namely for computing the intersection of  $\ell$  sets with a communication complexity of  $\mathcal{O}(\lambda \ell t \text{polylog } t)$ , when  $|\left(\cup_{i=1}^{\ell} X_i\right) \setminus \left(\cap_{i=1}^{\ell} X_i\right)| \leq t$ . For  $\ell = 2$ , the work of Badrinarayanan et al. is equivalent to two-party TPSI, but for  $\ell > 2$  their work requires the set intersection to not only be large, but additionally they require that the parties have less than  $t$  distinct elements outside the intersection among all sets. Both Branco, Döttling, and Pu as well as Badrinarayanan et al. show that one cannot do better than  $\Omega(\ell t)$  in their respective settings and provide, up to polylog factors, matching upper bounds based on fully homomorphic encryption.

All three works [GS19, BDP21, BMRR21] leave constructing asymptotically optimal multiparty protocols from other assumptions than the existence of fully homomorphic encryption as an open problem.

### 1.1 Applications of Threshold Private Set Intersection

As has been pointed out by the previous works, threshold private set intersection is not just an interesting theoretical object to study, but also has the potential to be useful in a variety of practical applications, where parties are only interested in the actual intersection if it is indeed large. In the biometric authentication setting, we have a biometric reading represented as a feature vector and a template. An authentication attempt can directly be discarded, if the reading has a small intersection with the template. In the setting of ride sharing or dating apps, users may not care to share their private data with each other, if they do not have a large intersection.

Even protocols for general private set intersection can benefit from more efficient TPSI protocols. Parties that would like to compute the intersection of their sets, can first execute a private intersection cardinality test protocol on thresholds  $2, 2^2, 2^4, \dots$  to determine the correct threshold and then compute the intersection using the TPSI protocol. Using this approach, leads to a general private set intersection protocol with a communication complexity that depends on the size of the output and not on the size of the inputs. This is in stark contrast

to the majority of existing works on PSI that usually have a communication complexity that is at least linear in the smallest set size.

## 1.2 Our Contribution

In this work, we present new protocols for computing the threshold private set intersection among  $\ell$  parties with a quasilinear rather than quadratic dependency on  $t$  from simple assumptions. More concretely, we construct protocols with a communication complexity of  $\mathcal{O}(\lambda^2 \ell t \text{polylog } t)$  bits. We follow the blueprint of Ghosh and Simkin [GS19] and tackle the problem by splitting it into two smaller problems. We first execute a private intersection cardinality test (PICT) protocol  $\Pi_{\ell\text{-pict}}^{n,t}(X_1, \dots, X_\ell)$  that checks, whether the given sets  $X_1, \dots, X_\ell$  have an intersection of size at least  $n - t$ . If they do, we can execute another protocol for computing the actual intersection in a communication efficient manner.

Computing the intersection, when it has already been established that it is indeed large enough, can be done generically from assumptions, like the existence of oblivious transfer or additively homomorphic encryption, with a close to optimal communication complexity of  $\tilde{\mathcal{O}}(\lambda \ell t)$  bits as has been shown by Ghosh and Simkin. Thus, the main challenge and the focus of this work is to construct communication efficient PICT protocols from simple assumptions, which output a single bit that indicates, whether the intersection is large enough.

Our main technical contribution is a transformation that takes any PICT protocol with secret shared outputs<sup>1</sup> and communication complexity  $\mathcal{O}(\lambda \ell \text{poly}(t))$  and transforms it into a new protocol that solves the same task, but has a communication complexity of only  $\mathcal{O}(\lambda^2 \ell t \text{polylog}(t))$ . An implication of this compiler is the existence of multiparty protocols with the above stated communication complexity from effectively any assumption that implies secure computation. The efficiency of a protocol that is given as input to our transformation only affects the constant in the  $\text{polylog}(t)$  exponent.

**Is This Stuff Practical?** We stress that the main focus of this work is to construct asymptotically efficient protocols from simpler assumptions. We hope that our work will eventually lead to practically efficient protocols, but we think that our current results achieving a communication complexity of  $\mathcal{O}(\lambda^2 \ell t \log^c t)$  bits for some  $c \geq 2$  are still slightly too inefficient for most reasonable real-world parameters. We leave constructing protocols with  $c \leq 1$  as an exciting open question for future work. Nonetheless, we view our work as a significant theoretical step towards more efficient protocols for threshold private set intersection.

## 1.3 Technical Overview

For the sake of this overview, let us focus on the two-party case. We would like to design a protocol that takes two sets  $X, Y \subset U$  from some universe  $U$  as input

<sup>1</sup> All existing protocols can easily be adapted to output secret shares of the output instead of the output itself.

and outputs a bit that indicates, whether  $|X \cap Y| \geq n - t$  or equivalently, whether the symmetric set difference  $|X \Delta Y| := |X \setminus Y \cup Y \setminus X| \leq 2t$ . Our main idea is to approach this problem via a divide and conquer strategy, i.e. to partition the sets  $X$  and  $Y$  into smaller sets  $X_1, \dots, X_t$  and  $Y_1, \dots, Y_t$  and then to perform a series of independent PICTs on each pair  $X_i$  and  $Y_i$  for  $i \in [t] := \{1, \dots, t\}$ .

More precisely, imagine we have random functions<sup>2</sup>  $H^i : U \rightarrow [t]$  for  $i \in [\epsilon]$  for some value  $\epsilon$  that take set elements as input and outputs values in  $[t]$ . Define  $X_i^j = \{x \mid x \in X \wedge H^j(x) = i\}$  and  $Y_i^j = \{y \mid y \in Y \wedge H^j(y) = i\}$  for  $i \in [t]$  and  $j \in [\epsilon]$  and observe that for all  $j \in [\epsilon]$

$$|X \Delta Y| = \sum_{i=1}^t \left| X_i^j \Delta Y_i^j \right|.$$

Consider some fixed  $j \in [\epsilon]$ . If  $|X \Delta Y| \leq 2t$ , then in expectation each pair of sets  $X_i^j$  and  $Y_i^j$  contains at most two elements in their symmetric set difference and one can show that (for a fixed  $j$ ) with a constant probability none of the pairs has a symmetric set difference that is larger than  $\mathcal{O}(\ln t)$ . It follows that when  $|X \Delta Y| \leq 2t$ , there must exist at least one  $j$  for which  $\left| X_i^j \Delta Y_i^j \right| \in \mathcal{O}(\ln t)$  for all  $i \in [t]$  with an overwhelming in  $\epsilon$  probability.

So how is this helpful? Imagine we were given access to an auxiliary functionality  $\mathcal{F}_{\Delta}^{n, \tilde{t}, v}$  that takes two sets as input and either returns a secret sharing of the size of their *exact* symmetric set difference or a secret sharing of some default value  $v$ , if the symmetric set difference is larger than  $\tilde{t} \approx \ln t$ . We can use  $\mathcal{F}_{\Delta}^{n, \tilde{t}, v}$  on each of the  $\epsilon t$  many subset pairs to obtain equally many secret shared values and then add all the values together that belong to inputs, which were partitioned using the same random partitioning function to get a total of  $\epsilon$  many secret shared sums. Each of those sums either equals the exact size of the symmetric set difference of  $X$  and  $Y$  or some value, which has  $v$  as a summand. By picking  $v = t + 1$ , we ensure that each sum containing  $v$  is larger than  $t$ . As the final step in our protocol, we run a generic secure computation protocol for checking, whether any of the  $\epsilon$  sums is at most  $t$  in which case we conclude that the inputs  $X$  and  $Y$  have a large enough intersection.

To make our protocol work, we still need to instantiate  $\mathcal{F}_{\Delta}^{n, \tilde{t}, v}$ . We show that this can be done from any PICT protocol with secret shared outputs for thresholds  $\tilde{t}$ . If the given protocol has a communication complexity of  $\mathcal{O}(\lambda \text{poly}(\tilde{t}))$  bits, then our instantiation of  $\mathcal{F}_{\Delta}^{n, \tilde{t}, v}$  has a communication complexity of  $\mathcal{O}(\lambda \tilde{t} \text{poly}(\tilde{t})) = \mathcal{O}(\lambda \ln t \text{polylog } t) = \mathcal{O}(\lambda \text{polylog } t)$ . Since our approach only relies on generic secure computation and existing PICT protocols, it follows that we can instantiate our constructions from assumptions that imply both of these cryptographic objects. As we will see, this means that we can instantiate our results from oblivious transfer or generic additively homomorphic encryption.

<sup>2</sup> Throughout the paper we will use random functions for the sake of simplicity, but we stress that all of our constructions and arguments work equally well with pseudorandom functions, where the key is known to all parties.

Our multiparty PICT protocols follows the same blueprint as the protocol outlined above, but need to overcome several other challenges. In the the two-party case we got away with just talking about the symmetric set difference, since an upper bound on that quantity directly translates into a lower bound on the set intersection size. In the multiparty setting this is not the case any longer and we will need to directly talk about the set intersection sizes in all the buckets instead. While it may sounds like an irrelevant change, it does introduce some small technical challenges that we will highlight in Sect. 4.

**Paper Outline.** In Sect. 2 we recall some basic preliminaries and define all the required notation that will be needed throughout the paper. In Sect. 3, we present our protocol for the two-party case. We stress that this *does not* asymptotically improve upon the state-of-the-art, which has a communication complexity of  $\mathcal{O}(\lambda t \text{ polylog } t)$  bits<sup>3</sup>. We do, however, believe that our two-party protocol highlights the main ideas of this work quite well, while avoiding some of the complexities that come from considering multiple parties. In Sect. 4 we present our multiparty protocol, which is the main technical contribution of this work.

## 2 Preliminaries

**Notation.** We write  $[n] = \{1, 2, \dots, n\}$ . Let  $\log x$  be the logarithm of  $x$  with base 2 and  $\ln x$  the one with base  $e$ . For convenience, we assume the natural numbers start at one, i.e.  $\mathbb{N} = \{1, 2, 3, \dots\}$ . Let  $\lambda$  be the computational and  $\epsilon$  the statistical security parameter and we assume that  $\epsilon/\lambda \in \mathcal{O}(1)$ . We write  $\mathbb{F}$  to denote a finite field of prime order and we assume that  $|\mathbb{F}| \geq 2^\epsilon$ . For parties  $P_1, \dots, P_\ell$  with inputs  $X_1, \dots, X_\ell$  that have oracle access to an ideal functionality  $\mathcal{F}$ , we write  $(b_1, \dots, b_\ell) \leftarrow \mathcal{F}(X_1, \dots, X_\ell)$  as a shorthand notation for each party  $i$  sending  $X_i$  to the ideal functionality and, once all inputs are received, receiving back output  $b_i$ . For a protocol  $\Pi$ , we write  $\text{CC}(\Pi)$  to denote the communication complexity of  $\Pi$ , i.e. the number of bits exchanged in one execution of the protocol.<sup>4</sup>

**Theorem 1 (Chernoff Bound).** *Let  $I_1, \dots, I_n$  be random variables with  $0 \leq I_i \leq 1$  for all  $i \in [n]$ . Define  $I = \sum_{i=1}^n I_i$  and let  $\mu = \mathbb{E}[I]$ . For any  $\delta \geq 1$ ,*

$$\Pr[I \geq (1 + \delta)\mu] \leq e^{-\frac{\delta\mu}{3}}.$$

---

<sup>3</sup> This communication complexity can be obtained, without using fully homomorphic encryption, by using the construction of Ghosh and Simkin [GS19] in combination with an observation due to Badrinarayanan et al. [BMRR21].

<sup>4</sup> We assume that the communication complexity is a deterministic function of the inputs and parameters of  $\Pi$ .

**Set Gymnastics.** Let  $U$  be the universe from which set elements will be sampled and let  $Z = (z_1, z_2, \dots)$  be an auxiliary (sorted) universe such that  $U \cap Z = \emptyset$ . We will use upper case letter for sets and lower case letters for their elements, e.g.  $S = \{s_1, \dots, s_n\}$ . For  $S \in U^n$  and function  $H : U \rightarrow [t]$ , we write  $(S_1, \dots, S_t) \leftarrow H(S)$  as a shorthand notation to specify the sets  $S_i = \{s \mid s \in S \wedge H(s) = i\}$ .

**Secret Sharing.** Let  $\text{Share}_n : \mathbb{F} \rightarrow \mathbb{F}^\ell$  be an  $\ell$ -out-of- $\ell$  secret sharing algorithm that takes  $v \in \mathbb{F}$  as input and outputs uniformly random  $v_1, \dots, v_\ell \in \mathbb{F}$ , such that  $v = \sum_{i=1}^\ell v_i$ . When an algorithm or a functionality outputs  $\text{Share}_\ell(v)$ , we mean that party  $i$  receives shares  $v_i$ .

### 2.1 Secure Multiparty Computation

We assume familiarity with standard secure computation notions in the standalone model (see [Lin17]). In this paper, we assume that all parties are pairwise connected via a synchronous network and authenticated private channels. Additionally the parties have access to a broadcast channel. We consider an adversary that can corrupt all but one parties passively.

### 2.2 Private Intersection Cardinality Testing

For the two-party case the functionality we are interested in is the  $\mathcal{F}_{\text{pict}}^{n,t}(X, Y)$  functionality shown in Fig. 1. It is helpful to note that for  $X$  and  $Y$  with  $n = |X| = |Y|$ , it holds that

$$|X \cap Y| \leq n - t \iff |X \Delta Y| > 2t,$$

which means that the functionality outputs a sharing of 1 for two sets of size  $n$  if and only if  $|X \cap Y| > n - t$ . The functionality  $\mathcal{F}_{\text{pict}}^{n,t}(X, Y)$  does allow for the input sets to be of unequal sizes smaller than  $n$  in which case the equivalence above does not hold. This is done for the sake of simplifying the presentation of our construction in the two-party case. The multiparty functionality will be introduced in Sect. 4 and it will require the input sets to be of the same size.

### 2.3 Some Auxiliary Functionalities

In the following, we define some helpful functionalities that will come in handy later on. They can be realized using any generic secure computation protocol and will not affect our communication complexities in any meaningful way.

The functionalities in Fig. 2 allow for comparing a secret shared input against a publicly known threshold and returning either the secret shared value or a default value. Both functionalities can be easily realized with communication complexities that are linear in their input length with standard secure computation tools.

```

Functionality  $\mathcal{F}_{\text{pict}}^{n,t}(X, Y)$ 


---


if  $|X| > n$  or  $|Y| > n$ 
    return  $\perp$ 
if  $|X \Delta Y| > t$ 
    return  $\text{Share}_2(0)$ 
else
    return  $\text{Share}_2(1)$ 
    
```

**Fig. 1.** Functionality takes two sets  $X$  and  $Y$  of size at most  $n$  as input and checks whether  $|X \Delta Y| \leq t$ .

<pre> <b>Functionality</b> <math>\mathcal{F}_{\text{cmp}}^{t,v}(r_1, r_2)</math> <hr/> <b>if</b> <math>r_1 + r_2 = t</math>     <b>return</b> <math>\text{Share}_2(v)</math> <b>else</b>     <b>return</b> <math>\text{Share}_2(r_1 + r_2)</math>                 </pre>	<pre> <b>Functionality</b> <math>\mathcal{F}_{\text{l-geq}}^{t,v}(r_1, \dots, r_\ell)</math> <hr/> <b>Compute</b> <math>s := \sum_{i=1}^{\ell} r_i</math> <b>if</b> <math>s \geq t</math>     <b>return</b> <math>\text{Share}_\ell(s)</math> <b>else</b>     <b>return</b> <math>\text{Share}_\ell(v)</math>                 </pre>
--	--

**Fig. 2.** Some useful private comparison function of secret shared inputs.

Functionality  $\mathcal{F}_{\text{l-vec-leq}}^{t,\epsilon}((s_1^1, \dots, s_1^\epsilon), \dots, (s_\ell^1, \dots, s_\ell^\epsilon))$  in Fig. 3 takes  $\epsilon$  many  $\ell$ -out-of- $\ell$  secret shared field elements as input and returns 1 if any one of them is smaller than  $t$  and 0 otherwise. This functionality can be realized using generic secure computation with a communication complexity of  $\mathcal{O}(\epsilon \ell |\mathbb{F}|)$  bits.

```

Functionality  $\mathcal{F}_{\text{l-vec-leq}}^{t,\epsilon}((s_1^1, \dots, s_1^\epsilon), \dots, (s_\ell^1, \dots, s_\ell^\epsilon))$ 


---


if  $\exists j \in [\epsilon] : \sum_{i=1}^{\ell} s_i^j \leq t$ 
    return  $\text{Share}_\ell(1)$ 
else
    return  $\text{Share}_\ell(0)$ 
    
```

**Fig. 3.** Functionality for checking, whether one of the  $\epsilon$  many secret shared inputs is at most  $t$ .



The functionality in Fig. 4 computes the minimum among a list of input values and returns that value in secret shared form.

<p><b>Functionality</b> <math>\mathcal{F}_{\ell\text{-min}}(d_1, \dots, d_\ell)</math></p> <hr style="border: 0.5px solid black;"/> <p><b>Compute</b> <math>d_{\min} := \min\{d_1, \dots, d_\ell\}</math></p> <p><b>return</b> <math>\text{Share}_\ell(d_{\min})</math></p>
---

**Fig. 4.** Functionality for computing a secret sharing of the minimum among a set of inputs.

The functionality in Fig. 5 checks whether at least one of multiple secret shared values is within a given interval.

### 3 The Two-Party Divide-and-Conquer Approach

In this section, we will focus on the two-party case for the sake of presenting our main ideas in a simplified setting.

Let us begin with a simple lemma, which states that one can partition sets  $X$  and  $Y$  into smaller sets and compute the size of their symmetric set difference in a divide-and-conquer fashion.

<p><b>Functionality</b> <math>\mathcal{F}_{\ell\text{-vec-intv}}^{n,t,\epsilon}((s_1^1, \dots, s_1^\epsilon), \dots, (s_\ell^1, \dots, s_\ell^\epsilon))</math></p> <hr style="border: 0.5px solid black;"/> <p><b>if</b> <math>\exists j \in [\epsilon] : n - t \leq \sum_{i=1}^{\ell} s_i^j \leq n</math></p> <p style="padding-left: 20px;"><b>return</b> <math>\text{Share}_\ell(1)</math></p> <p><b>else</b></p> <p style="padding-left: 20px;"><b>return</b> <math>\text{Share}_\ell(0)</math></p>
--

**Fig. 5.** Functionality for checking, whether at least one of  $\epsilon$  many secret shared input values is in between  $n - t$  and  $n$ .

**Lemma 1.** *Let  $X, Y \subset U$ , let  $t \in \mathbb{N}$ , and let  $H : U \rightarrow [t]$  be an arbitrary function. For  $(X_1, \dots, X_t) \leftarrow H(X)$  and  $(Y_1, \dots, Y_t) \leftarrow H(Y)$ , it holds that*

$$|X \Delta Y| = \sum_{i=1}^t |X_i \Delta Y_i|.$$

*Proof.* Consider two arbitrary sets  $X, Y \subset U$ . We observe that  $|X \Delta Y| = |X \setminus Y| + |Y \setminus X|$ . If  $v \in X \setminus Y$ , then there exists an index  $i \in [t]$  such that  $v \in X_i \setminus Y_i$  and since  $X_i \cap X_j = \emptyset$  for any  $j \neq i$ , it holds that  $i$  is unique. The other way round, for any  $i \in [t]$  and any  $v \in X_i \setminus Y_i$ , it holds that  $v \in X \setminus Y$ . Thus

$$|X \setminus Y| = \sum_{i=1}^t |X_i \setminus Y_i|$$

and by symmetry of the above argument

$$\begin{aligned} |X \Delta Y| &= |X \setminus Y| + |Y \setminus X| \\ &= \sum_{i=1}^t |X_i \setminus Y_i| + \sum_{i=1}^t |Y_i \setminus X_i| \\ &= \sum_{i=1}^t |X_i \setminus Y_i| + |Y_i \setminus X_i| = \sum_{i=1}^t |X_i \Delta Y_i|. \end{aligned}$$

□

Next, we observe that, if the symmetric set difference of  $X$  and  $Y$  is at most  $t$ , then the symmetric set difference of each pair of subsets  $X_i$  and  $Y_i$  for  $i \in [t]$  is in  $\mathcal{O}(\ln t)$  with a constant probability.

**Lemma 2.** *Let  $n, t, \tilde{t} \in \mathbb{N}$  with  $t < n$  and  $\tilde{t} \geq 1 + 3 \ln 2t$ . Let  $H : U \rightarrow [t]$  be a random function, and let  $X, Y \in U^n$ . If  $|X \Delta Y| \leq t$ , then for  $(X_1, \dots, X_t) \leftarrow H(X)$  and  $(Y_1, \dots, Y_t) \leftarrow H(Y)$  it holds that*

$$\Pr [\exists i \in [t] : |X_i \Delta Y_i| \geq \tilde{t}] \leq 1/2,$$

where the probability is taken over the random choice of  $H$ .

*Proof.* Assume that  $|X \Delta Y| \leq t$ . For all  $i \in [t]$ , it holds that  $X_i \Delta Y_i = \{v \mid v \in X \Delta Y \wedge H(v) = i\} \subset X \Delta Y$ . Fix one bucket  $j$  and let  $I_v$  be the indicator variable for whether  $v \in X \Delta Y$  landed in bucket  $j$  or not. For

$$\mathbb{E}[|X_j \Delta Y_j|] = \mathbb{E} \left[ \sum_{v \in X \Delta Y} I_v \right] = \sum_{v \in X \Delta Y} 1/t \leq 1$$

we get by Chernoff bound that

$$\Pr [|X_j \Delta Y_j| \geq 1 + 3 \ln 2t] \leq e^{-\frac{3 \ln 2t}{3}} = 1/2t,$$

where the probability is taken over the random choice of the function  $H$ . The statement follows by union bounding over all  $t$  buckets. □

Now, if  $|X \Delta Y| \leq t$  and we partition sets  $X$  and  $Y$  not once, but  $\epsilon$  many times, then we are guaranteed with overwhelming probability that at least one of those partitions has no bucket that contains more than  $\mathcal{O}(\ln t)$  elements.

**Theorem 2.** *Let  $n, t, \tilde{t} \in \mathbb{N}$  with  $t < n$  and  $\tilde{t} \geq 1 + 3 \ln 2t$ . For each  $i \in [\epsilon]$ , let  $H^i : U \rightarrow [t]$  be a random function. Let  $X, Y \in U^n$  be two sets of size  $n$  and  $(X_1^i, \dots, X_t^i) \leftarrow H^i(X)$  and  $(Y_1^i, \dots, Y_t^i) \leftarrow H^i(Y)$  for  $i \in [\epsilon]$ . If  $|X \Delta Y| \leq t$ , then*

$$\Pr \left[ \exists i_1, \dots, i_\epsilon \in [t] : \left| X_{i_j}^j \Delta Y_{i_j}^j \right| \geq \tilde{t} \forall j \in [\epsilon] \right] \leq 2^{-\epsilon},$$

where the probability is taken over the random choice of  $H^1, \dots, H^\epsilon$ .

*Proof.* Assume  $|X \Delta Y| \leq t$ , then

$$\begin{aligned} & \Pr \left[ \exists i_1, \dots, i_\epsilon \in [t] : \left| X_{i_j}^j \Delta Y_{i_j}^j \right| \geq 1 + 3 \ln 2t \forall j \in [\epsilon] \right] \\ &= \prod_{j=1}^{\epsilon} \Pr \left[ \exists i_j \in [t] : \left| X_{i_j}^j \Delta Y_{i_j}^j \right| \geq 1 + 3 \ln 2t \right] \\ &\leq \prod_{j=1}^{\epsilon} 1/2 = 2^{-\epsilon}, \end{aligned}$$

where the last inequality follows from Lemma 2. □

From the above it now follows that, if there exists at least a single bucket in each of the  $\epsilon$  partitions, which contains more than  $1 + 3 \ln 2t$  elements of the symmetric set difference, then we can conclude that  $|X \Delta Y| > t$  with overwhelming probability.

**Corollary 1.** *Let  $n, t, \tilde{t} \in \mathbb{N}$  with  $t < n$  and  $\tilde{t} \geq 1 + 3 \ln 2t$ . For each  $i \in [\epsilon]$ , let  $H^i : U \rightarrow [t]$  be a random function. Let  $X, Y \in U^n$  be two sets of size  $n$  and  $(X_1^i, \dots, X_t^i) \leftarrow H^i(X)$  and  $(Y_1^i, \dots, Y_t^i) \leftarrow H^i(Y)$  for  $i \in [\epsilon]$ . If there exist indices  $i_1, \dots, i_\epsilon \in [t]$ , such that for all  $j \in [\epsilon]$  it holds that  $\left| X_{i_j}^j \Delta Y_{i_j}^j \right| \geq \tilde{t}$ , then*

$$\Pr[|X \Delta Y| > t] \geq 1 - 2^{-\epsilon},$$

where the probability is taken over the random choices of  $H^1, \dots, H^\epsilon$ .

**Functionality**  $\mathcal{F}_{\Delta}^{n,t,v}(X, Y)$

---

**if**  $|X| > n$  **or**  $|Y| > n$   
**return**  $\perp$

**if**  $|X \Delta Y| > t$   
**return**  $\text{Share}_2(v)$

**else**  
**return**  $\text{Share}_2(|X \Delta Y|)$

**Fig. 6.** Functionality for computing the exact symmetric set difference, if it is smaller than  $t$ , of sets  $X$  and  $Y$  with elements from  $U$ . The sets  $X$  and  $Y$  may be of different sizes, but neither of them is larger than  $n$ .

Armed with the above observations, we are now ready to present our construction. The description of our protocol makes use of an ideal functionality  $\mathcal{F}_{\Delta}^{n,t,v}$  (see Fig. 6) that takes two sets as input and either returns a secret sharing of their symmetric set difference or returns a sharing of some value  $v$ . The sets may be of different sizes, but are both not larger than  $n$ . We want to highlight that allowing for input sets of unequal size is only possible, because we are currently talking about the symmetric set difference. Looking ahead, we will be directly talking about the size of the intersection in the multiparty protocols in Sect. 4 and therefore we will need to take care of making the sets be of the correct and same size. We show how to instantiate  $\mathcal{F}_{\Delta}^{n,t,v}$  in Sect. 3.1

**Theorem 3.** *Let  $n, t, \tilde{t} \in \mathbb{N}$  with  $n > t$  and  $\tilde{t} = 1 + 3 \ln 2t$ . The protocol  $\Pi_{\text{pict}}$  depicted in Fig. 7 securely realizes  $\mathcal{F}_{\text{pict}}^{n,t}$  using  $\epsilon \cdot t$  calls to  $\mathcal{F}_{\Delta}^{n,\tilde{t},t+1}$  and one call to  $\mathcal{F}_{\text{vec-cmp}}^{t,\epsilon}$ .*

*Proof.* We prove correctness and privacy separately.

*Correctness.* If there exists indices  $j_1, \dots, j_{\epsilon} \in [t]$  such that  $|X_{j_i}^i \Delta Y_{j_i}^i| > \tilde{t}$  for all  $j \in [\epsilon]$ , then by Corollary 1 we know that  $|X \Delta Y| > t$  with overwhelming probability and thus the output of  $\Pi_{\text{pict}}^{n,t}(X, Y)$  should be a secret sharing of 0. We observe that for these indices it holds that  $r_{j_i}^i + s_{j_i}^i = t + 1$  and thus for all  $i \in [\epsilon]$  it holds that  $r^i + s^i > t$ . Therefore  $r + s = 0$

<b>Construction</b> $\Pi_{\text{pict}}^{n,t}(X, Y)$	
1:	<b>for</b> $i \in [\epsilon]$ :
2:	<b>Alice:</b> $(X_1^i, \dots, X_t^i) \leftarrow H^i(X)$
3:	<b>Bob:</b> $(Y_1^i, \dots, Y_t^i) \leftarrow H^i(Y)$
4:	<b>for</b> $j \in [t]$ :
5:	$(r_j^i, s_j^i) \leftarrow \mathcal{F}_{\Delta}^{n,\tilde{t},t+1}(X_j^i, Y_j^i)$
6:	<b>Alice:</b> $r^i := \sum_{j=1}^t r_j^i$
7:	<b>Bob:</b> $s^i := \sum_{j=1}^t s_j^i$
8:	$(r, s) \leftarrow \mathcal{F}_{2\text{-vec-leq}}^{t,\epsilon}((r^1, \dots, r^{\epsilon}), (s^1, \dots, s^{\epsilon}))$
9:	<b>return</b> $(r, s)$

**Fig. 7.** Protocol for private intersection cardinality testing.

If on the other hand there exists an  $i \in [\epsilon]$ , such that  $|X_j^i \Delta Y_j^i| \leq \tilde{t}$  for all  $j \in [t]$ , then

$$r^i + s^i = \sum_{j=1}^t r_j^i + s_j^i = \sum_{j=1}^t |X_j^i \Delta Y_j^i| \stackrel{\text{(Lemma 1)}}{=} |X \Delta Y|$$

and thus the  $\mathcal{F}_{\text{vec-cmp}}^{t,\epsilon}$  outputs a sharing of 1 if and only if  $|X \Delta Y| \leq t$ .

*Privacy.* Without loss of generality assume that Alice is corrupted. At each step of the protocol, she only sees one share of freshly independent secret shared values returned by the ideal functionalities. Her view can simply be simulated by providing her shares of independent secret sharings of 0 instead of the real values. The indistinguishability of Alice’s simulated view follows from the indistinguishability of the secret sharing scheme. □

### 3.1 Instantiating $\mathcal{F}_{\Delta}^{n,t,v}$

To instantiate  $\mathcal{F}_{\Delta}^{n,t,v}$ , we simply use  $\mathcal{F}_{\text{pict}}^{n,i}$  once for each threshold  $i \in [t]$  and then accumulate the result.

**Theorem 4.** *Let  $n, t \in \mathbb{N}$  with  $n > t$  and  $v \in \mathbb{F}$ . The protocol  $\Pi_{\Delta}^{n,t,v}$  depicted in Fig. 8 securely implements  $\mathcal{F}_{\Delta}^{n,t,v}$  using one call to  $\mathcal{F}_{\text{pict}}^{n,i}$  for each  $i \in [t]$ .*

**Construction  $\Pi_{\Delta}^{n,t,v}(X, Y)$**

---

- 1: **for**  $i \in [t]$  :
- 2:      $(r_i, s_i) \leftarrow \mathcal{F}_{\text{pict}}^{n,i}(X, Y)$
- 3: **Alice:**  $r := t - \sum_{j=1}^t r_j$
- 4: **Bob:**  $s := - \sum_{j=1}^t s_j$
- 5:  $(d_1, d_2) \leftarrow \mathcal{F}_{\text{cmp}}^{t,v}(r, s)$
- 6: **return**  $(d_1, d_2)$

**Fig. 8.** Protocol  $\Pi_{\Delta}^{n,t,v}$  realizing  $\mathcal{F}_{\Delta}^{n,t,v}$ .

*Proof.* For correctness, we observe that  $\mathcal{F}_{\text{pict}}^{n,i}(X, Y)$  outputs a sharing of 1, when  $|X \Delta Y|$  is at most  $i$ . Thus, for  $|X \Delta Y| \leq t$ , we have that  $(r, s)$  is a secret sharing of exactly  $|X \Delta Y|$  and if  $|X \Delta Y| > t$ , then  $(r, s)$  is a secret sharing of  $t$ .

For seeing that the protocol is secure, assume that Alice is corrupted. To simulate the responses of  $\mathcal{F}_{\text{pict}}^{n,i}(X, Y)$ , we add shares of a secret sharing of 0 to her view. Given the output of the functionality, we secret share that value and add one share to Alice’s view. It is straightforward to see that this perfectly simulates Alice’s view in the real world, which completes the proof.  $\square$

To instantiate our overall protocol, we now need to instantiate the  $\mathcal{F}_{\text{pict}}^{n,t}$  functionality that is being used inside of  $\Pi_{\Delta}^{n,t,v}$ . Formally, the two-party protocols of Ghosh and Simkin [GS19] require the input sets to be of the same size. Their protocols, however, work equally well for sets of different sizes and thus can be used to instantiate our functionality  $\mathcal{F}_{\text{pict}}^{n,t}$ . Internally, their work relies on a protocol for securely computing the determinant of a secret shared matrix. They instantiate that protocol with a communication complexity of  $\mathcal{O}(\lambda t^2 \text{polylog}(t))$  via additively homomorphic encryption, but using a protocol for computing that determinant by Cramer and Damgård [CD01], one can instantiate the protocol of Ghosh and Simkin with communication complexity  $\mathcal{O}(\lambda \ln^3 t)$  from generic secure computation. It follows that our result can be instantiated from any assumption, such as the existence of additively homomorphic encryption or oblivious transfer, that implies secure computation.

In our instantiation, we have  $et$  buckets and for each of them we execute the protocol of Ghosh and Simkin  $\mathcal{O}(\ln t)$  times with a threshold of  $\mathcal{O}(\ln t)$ . Thus we get the following corollaries.

**Corollary 2.** *Assuming the existence of oblivious transfer (or additively homomorphic encryption), there exists a constant-round protocol for securely computing the two-party private intersection cardinality test for threshold  $t$  with communication complexity of  $\mathcal{O}(\epsilon^2 t \text{polylog } t)$  bits.*

Combining the results in our paper with the protocols for actually computing the intersection, once it is known that it is large enough, from by Ghosh and Simkin [GS19], we get the following result.

**Corollary 3.** *Assuming the existence of oblivious transfer (or additively homomorphic encryption), there exists a constant-round protocol for threshold private set intersection among two parties with threshold  $t$  with communication complexity of  $\mathcal{O}(\epsilon^2 t \text{polylog } t)$  bits.*

## 4 The Multiparty Case

We now proceed to present our protocol for the multiparty case, which follows the blueprint from Sect. 3, but needs to overcome several additional challenges. The functionality that we would like to realize in this section is depicted in Fig. 9.

```

Functionality  $\mathcal{F}_{\ell\text{-pict}}^{n,t}(X_1, \dots, X_\ell)$ 
if  $|X_1| \neq n$  or ... or  $|X_\ell| \neq n$ 
  return  $\perp$ 
if  $\left| \bigcap_{i=1}^{\ell} X_i \right| \geq n - t$ 
  return  $\text{Share}_\ell(1)$ 
else
  return  $\text{Share}_\ell(0)$ 

```

**Fig. 9.** Functionality for multiparty private intersection cardinality testing among sets of size exactly  $n$ .

In the two-party case we got away with talking about the set difference as a surrogate for the size of the set intersection due to the equivalence of intersection size and size of the symmetric set difference that is pointed out in Sect. 2.2. In the multiparty case, we make no such assumption.

To call a protocol for computing the size of the intersection in each bucket among  $\ell$  parties, we will now ensure that the sets in each bucket are of the same size. We achieve this by padding sets with elements from an (ordered) auxiliary universe  $Z = \{z_1, z_2, \dots\}$  with  $Z \cap U = \emptyset$ . For  $n, b \in \mathbb{N}$  with  $b > n$  and any set  $X \in U^n$ , we define  $\text{Pad}(X, b) := X \cup \{z_i \mid i \in [n - b]\}$ . Lemma 3 shows the relationship between the size of the intersection among padded sets and unpadded sets.

**Lemma 3.** *Let  $b \in \mathbb{N}$  and let  $X_1, \dots, X_\ell \subset U$  with  $|X_i| \leq b$  for all  $i \in [\ell]$ . Let  $d_i := ||X_i| - b|$  for  $i \in [\ell]$ . Then*

$$\left| \bigcap_{i=1}^{\ell} X_i \right| = \left| \bigcap_{i=1}^{\ell} \text{Pad}(X_i, b) \right| - \min(d_1, \dots, d_\ell)$$

*Proof.* Define  $W_i = \text{Pad}(X_i, b) \setminus X_i$  for  $i \in [\ell]$ . We observe that

$$\left| \bigcap_{i=1}^{\ell} \text{Pad}(X_i, b) \right| = \left| \bigcap_{i=1}^{\ell} (X_i \cup W_i) \right| = \left| \bigcap_{i=1}^{\ell} X_i \right| + \left| \bigcap_{i=1}^{\ell} W_i \right| = \left| \bigcap_{i=1}^{\ell} X_i \right| + \min(d_1, \dots, d_\ell),$$

where the second equality follows from the fact that  $Z \cap U = \emptyset$  and thus the lemma statement follows. □

The following Lemma can be seen as a generalization of Lemma 1 to the multiparty case. On an intuitive level, it states that a lower bound on the size of the intersection of  $\ell$  sets translates into a lower bound on the cumulative size of the intersections in each buckets

**Lemma 4.** Let  $n, \ell, t \in \mathbb{N}$  with  $t < n$ , let  $H : U \rightarrow [t]$  be a random function, let  $X_1, \dots, X_\ell \in U^n$ , and  $(X_{i,1}, \dots, X_{i,t}) \leftarrow H(X_i)$  for  $i \in [\ell]$ . It holds that

$$\left| \bigcap_{i=1}^{\ell} X_i \right| \geq n - t$$

if and only if

$$\sum_{k=1}^t \left| X_{j,k} \setminus \bigcap_{i=1}^{\ell} X_{i,k} \right| \leq t, \forall j \in [\ell].$$

*Proof.* Let  $W_{j,k} := X_{j,k} \setminus \bigcap_{i=1}^{\ell} X_{i,k}$  for  $k \in [t]$  and  $j \in [\ell]$ . We observe that for each pair  $k, k' \in [t]$ , it holds that  $W_{j,k} \cap W_{j,k'} = \emptyset$  and thus

$$\left| X_j \setminus \bigcap_{i=1}^{\ell} X_i \right| = \sum_{k=1}^t \left| X_{j,k} \setminus \bigcap_{i=1}^{\ell} X_{i,k} \right|.$$

The statement follows from the fact that

$$\left| \bigcap_{i=1}^{\ell} X_i \right| \geq n - t,$$

if and only if

$$\left| X_j \setminus \bigcap_{i=1}^{\ell} X_i \right| \leq t, \forall j \in [\ell].$$

□

Similarly to Theorem 2, we will now show that, if the intersection is large enough, then there exists an index  $j \in [\epsilon]$  such that the partitioning with  $H^j$  will not result in any one party having too many elements in a single bucket that do not belong to that buckets intersection.

**Theorem 5.** Let  $n, \ell, t, \tilde{t} \in \mathbb{N}$  with  $t < n$  and  $\tilde{t} \geq 1 + 3 \ln(2t\ell)$ . For each  $j \in [\epsilon]$ , let  $H^j : U \rightarrow [t]$  be a random function. Let  $X_1, \dots, X_\ell \in U^n$  be sets of size  $n$  and  $(X_{i,1}^j, \dots, X_{i,t}^j) \leftarrow H^j(X_i)$  for  $j \in [\epsilon]$  and  $i \in [\ell]$ . If

$$\left| \bigcap_{i=1}^{\ell} X_i \right| \geq n - t,$$

then

$$\Pr \left[ \exists k_1, \dots, k_\epsilon \in [t] : \left| X_{i_j, k_j}^j \setminus \bigcap_{m=1}^{\ell} X_{m, k_j}^j \right| \geq \tilde{t} \forall j \in [\epsilon] \right] \leq 2^{-\epsilon},$$

where the probability is taken over the random choice of  $H^1, \dots, H^\epsilon$ .



*Proof.* Let  $W_i := X_i \setminus \left(\bigcap_{m=1}^{\ell} X_m\right)$  for  $i \in [\ell]$ . Assume  $\left|\bigcap_{m=1}^{\ell} X_m\right| > n - t$ , then for each  $i \in [\ell]$ , it holds that  $|W_i| \leq t$ . Fix some  $i \in [\ell]$ ,  $j \in [\epsilon]$ ,  $k \in [t]$  and consider  $X_{i,k}^j$ , where  $(X_{i,1}^j, \dots, X_{i,t}^j) \leftarrow H^j(X_i)$ . For  $v \in W_i$ , let  $I_v$  be the indicator variable for whether  $v \in X_{i,k}^j$  or not. Then,

$$\mathbb{E} \left[ \sum_{v \in W_i} I_v \right] = \sum_{v \in W_i} 1/t \leq 1$$

and thus by Chernoff bound

$$\Pr \left[ \sum_{v \in W_i} I_v \geq 1 + 3 \ln(2t\ell) \right] \leq e^{-\frac{3 \ln(2t\ell)}{3}} = 1/2t\ell.$$

By union bound over all  $t$  buckets and all  $\ell$  sets, we can thus conclude that

$$\Pr \left[ \exists k \in [t], i \in [\ell] : \left| X_{i,k}^j \setminus \left(\bigcap_{m=1}^{\ell} X_{m,k}^j\right) \right| > 1 + 3 \ln(2t\ell) \right] \leq 1/2.$$

It follows that

$$\begin{aligned} & \Pr \left[ \exists \begin{matrix} k_1, \dots, k_{\epsilon} \in [t] \\ i_1, \dots, i_{\epsilon} \in [\ell] \end{matrix} : \left| X_{i_j, k_j}^j \setminus \bigcap_{m=1}^{\ell} X_{m, k_j}^j \right| \geq 1 + 3 \ln(2t\ell) \forall j \in [\epsilon] \right] \\ &= \prod_{j=1}^{\epsilon} \Pr \left[ \exists k_j \in [t], i_j \in [\ell] : \left| X_{i_j, k_j}^j \setminus \bigcap_{m=1}^{\ell} X_{m, k_j}^j \right| \geq 1 + 3 \ln(2t\ell) \forall j \in [\epsilon] \right] \leq 2^{-\epsilon}. \end{aligned}$$

□

**Corollary 4.** Let  $n, t, \tilde{t} \in \mathbb{N}$  with  $t < n$  and  $\tilde{t} \geq 1 + 3 \ln(2t\ell)$ . For each  $j \in [\epsilon]$ , let  $H^j : U \rightarrow [t]$  be a random function. Let  $X_1, \dots, X_{\ell} \in U^n$  be sets of size  $n$  and  $(X_{i,1}^j, \dots, X_{i,t}^j) \leftarrow H^j(X_i)$  for  $j \in [\epsilon]$  and  $i \in [\ell]$ . If there exist indices  $k_1, \dots, k_{\epsilon} \in [t]$  and  $i_1, \dots, i_{\epsilon} \in [\ell]$ , such that for all  $j \in [\epsilon]$  it holds that

$$\left| X_{i_j, k_j}^j \setminus \bigcap_{m=1}^{\ell} X_{m, k_j}^j \right| \geq \tilde{t},$$

then

$$\Pr \left[ \left| \bigcap_{i=1}^{\ell} X_i \right| < n - t \right] \geq 1 - 2^{-\epsilon}$$

where the probability is taken over the random choices of  $H^1, \dots, H^{\epsilon}$ .

When partitioning a set into several subsets randomly, one cannot guarantee that all subsets will be of the same size. This is problematic, since we would like to view each party’s buckets as inputs to smaller instances of a private multi-party intersection cardinality testing problem. That is, if the different parties

have inputs of different (secret) sizes, then it is not clear what it means for an intersection to be large enough. For this reason, each party will not directly input its subset, but rather a padded version of it. Since the communication complexities of our protocols never depend on the input sizes, we simply pad each bucket to its maximum size.

**Lemma 5.** *Let  $n, \ell, t, \tilde{t}, b \in \mathbb{N}$  with  $t \leq \tilde{t} < n$  and  $b := n$  and  $H : U \rightarrow [t]$  be a random function. Let  $X_1, \dots, X_\ell \in U^n$  be sets of size  $n$  and  $(X_{i,1}, \dots, X_{i,t}) \leftarrow H(X_i)$  for  $i \in [\ell]$ .*

If

$$\left| X_{j,k} \setminus \bigcap_{i=1}^{\ell} X_{i,k} \right| < \tilde{t}, \forall j \in [\ell], k \in [t], \tilde{t} \in \mathbb{N}$$

then

$$\left| \text{Pad}(X_{j,k}, b) \setminus \bigcap_{i=1}^{\ell} \text{Pad}(X_{i,k}, b) \right| < \tilde{t}, \forall j \in [\ell], k \in [t].$$

*Proof.* Fix some  $k \in [t]$  and define  $t_j := \left| X_{j,k} \setminus \bigcap_{i=1}^{\ell} X_{i,k} \right|$  for  $j \in [\ell]$ . Observe that

$$t_j + \left| \bigcap_{i=1}^{\ell} X_{i,k} \right| + |\text{Pad}(X_{j,k}, b) \setminus X_{j,k}| = b$$

and thus for  $j, j' \in [\ell]$  we have

$$\begin{aligned} t_j + \left| \bigcap_{i=1}^{\ell} X_{i,k} \right| + |\text{Pad}(X_{j,k}, b) \setminus X_{j,k}| &= t_{j'} + \left| \bigcap_{i=1}^{\ell} X_{i,k} \right| + |\text{Pad}(X_{j',k}, b) \setminus X_{j',k}| \\ \iff t_j + |\text{Pad}(X_{j,k}, b) \setminus X_{j,k}| - |\text{Pad}(X_{j',k}, b) \setminus X_{j',k}| &= t_{j'} \end{aligned}$$

Now consider index  $j'$  such that  $|X_{j',k}| \geq |X_{j,k}|$  for any other  $j \in [\ell]$ . For that index  $j'$  it holds that  $\text{Pad}(X_{j',k}, b) \setminus X_{j',k} \subseteq \text{Pad}(X_{j,k}, b) \setminus X_{j,k}$ . In other words, this means that the elements that were used for padding the bucket belonging to party  $j'$  will be exactly the added elements in the intersection. Thus, for any other  $j$  the number of elements not in the intersection will be  $t_j + |\text{Pad}(X_{j,k}, b) \setminus X_{j,k}| - |\text{Pad}(X_{j',k}, b) \setminus X_{j',k}|$ . Now if by assumption  $t_{j'} < \tilde{t}$ , then  $t_j + |\text{Pad}(X_{j,k}, b) \setminus X_{j,k}| - |\text{Pad}(X_{j',k}, b) \setminus X_{j',k}| < \tilde{t}$ .  $\square$

Combining all of the above observations, we now get the following lemma.

**Theorem 6.** *Let  $n, \ell, t, \tilde{t}, b \in \mathbb{N}$  with  $t < n$ ,  $\tilde{t} \geq 1 + 3 \ln(2t\ell)$  and let  $b = n$ . For each  $j \in [\epsilon]$ , let  $H^j : U \rightarrow [t]$  be a random function. Let  $X_1, \dots, X_\ell \in U^n$  be sets of size  $n$  and  $(X_{i,1}^j, \dots, X_{i,t}^j) \leftarrow H^j(X_i)$  for  $j \in [\epsilon]$  and  $i \in [\ell]$ . If*

$$\left| \bigcap_{i=1}^{\ell} X_i \right| \geq n - t,$$

then

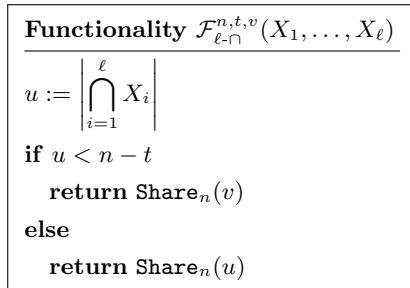
$$\Pr \left[ \forall k \in [t], \forall i \in [\ell], \exists j \in [\epsilon] : \left| \text{Pad}(X_{i,k}^j, b) \setminus \bigcap_{m=1}^{\ell} \text{Pad}(X_{m,k}^j, b) \right| < \tilde{t} \right] \geq 1 - 2^{-\epsilon},$$

where the probability is taken over the random choice of  $H^1, \dots, H^\epsilon$ .

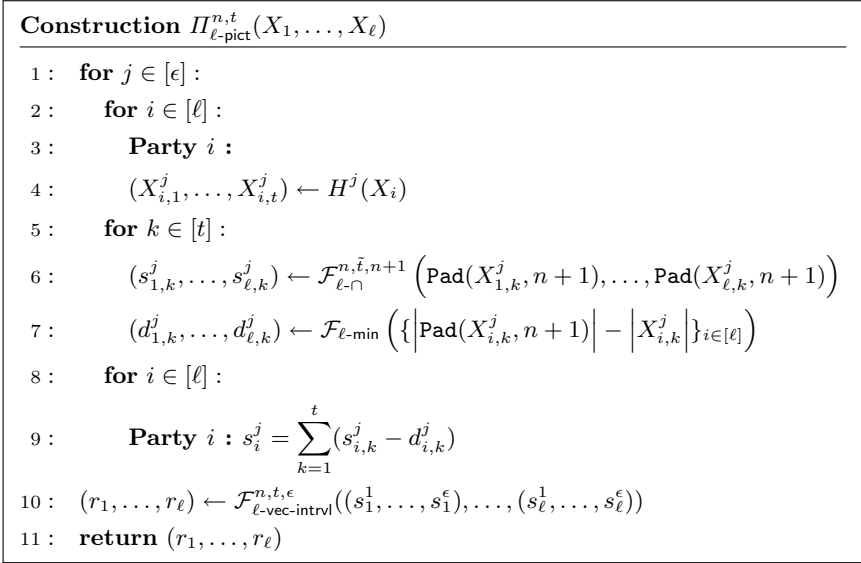
*Proof.* By Theorem 5 we know that if  $\left| \bigcap_{i=1}^{\ell} X_i \right| \geq n - t$ , then for all  $i \in [\ell]$  and  $k \in [t]$ , there exist  $j \in [\epsilon]$ , such that  $\left| X_{i,k}^j \setminus \bigcap_{m=1}^{\ell} X_{m,k}^j \right| < \tilde{t}$  with overwhelming probability. Also from Lemma 5 we know  $\left| \text{Pad}(X_{i,k}^j, b) \setminus \bigcap_{m=1}^{\ell} \text{Pad}(X_{m,k}^j, b) \right| < \tilde{t}$  in that case. The proof directly follows from these two observations. □

Armed with the insights from above, we are now ready to present our multiparty construction. We will assume that we are given access to an ideal functionality  $\mathcal{F}_{\ell-\cap}^{n,t,v}$  as depicted in Fig. 10 and we will show how to concretely instantiate it in Sect. 4.1. We also use two other simple functionalities  $\mathcal{F}_{\ell-\min}$  and  $\mathcal{F}_{\ell-\text{vec-intrvl}}^{n,t,\epsilon}$  in our protocol, which is described in Fig. 4 and Fig. 5 respectively. Note that these functionalities can be implemented using any generic MPC protocol with communication complexities that are independent of the initial set size  $n$  or threshold  $t$ .

In Fig. 11 we instantiate the protocol for multiparty private cardinality testing. Similar to the two-party case, here all the parties throw their set elements into  $t$  buckets and then run separate instances of cardinality test protocol among those buckets with a threshold parameter  $\tilde{t}$ , as stated in Theorem 6.



**Fig. 10.** Functionality for computing the number of elements in the intersection.



**Fig. 11.** Protocol for multiparty private intersection cardinality testing.

**Theorem 7.** *Let  $n, t, \tilde{t} \in \mathbb{N}$  with  $n > t$  and  $\tilde{t} \geq 1 + 3 \ln(2t\ell)$ . The protocol  $\Pi_{\ell\text{-pict}}$  depicted in Fig. 11 securely realizes  $\mathcal{F}_{\ell\text{-pict}}^{n,t}$  using  $\epsilon \cdot t$  calls to  $\mathcal{F}_{\ell\text{-}\cap}^{b,\tilde{t},n+1}$  and  $\mathcal{F}_{\ell\text{-min}}$  and one call to  $\mathcal{F}_{\ell\text{-vec-intrvl}}^{n,t,\epsilon}$ .*

*Proof.* We prove correctness and privacy separately.

*Correctness.* If  $|\bigcap_{i=1}^\ell X_i| < n - t$ , then by Theorem 6, we know that no bucket will overflow with an overwhelming probability in which case the protocol computes the size of the intersection correctly. If  $|\bigcap_{i=1}^\ell X_i| \geq n - t$ , then two things can happen. Either there will exist indices  $k_1, \dots, k_\epsilon \in [t]$  and  $i_1, \dots, i_\epsilon \in [\ell]$ , such that for all  $j \in [\epsilon]$  it holds that  $|X_{i_j, k_j}^j \setminus \bigcap_{m=1}^\ell X_{m, k_j}^j| \geq \tilde{t}$ . In this case, by Corollary 4, we know that the intersection was too small with an overwhelming probability. By construction, each sum of secret shared values per partitioning will contain a summand of  $n + 1$  and thus the sums will always be larger than  $n$  in which case  $\mathcal{F}_{\ell\text{-vec-intrvl}}^{n,t,\epsilon}$  returns 0 as desired.

Otherwise, the intersection is too small, but no bucket overflows. Since no bucket overflows, the parties correctly compute a secret sharing of the actual intersection and thus  $\mathcal{F}_{\ell\text{-vec-intrvl}}^{n,t,\epsilon}$  will produce the correct output of the computation.

*Privacy.* Without loss of generality assume that  $P_1, \dots, P_{\ell-1}$  are corrupted. We observe that the only communication the parties have during a protocol execution is through oracle calls. Each oracle call returns a fresh secret sharing of some random value and the parties always receives a subset of shares that is insufficient to reconstruct. To simulate the corrupted parties' views, we simply return shares of fresh secret sharings of 0 for each oracle call. □

#### 4.1 Instantiating $\mathcal{F}_{\ell-\cap}^{n,t,v}$

To instantiate  $\mathcal{F}_{\ell-\cap}^{n,t,v}$ , we use  $\mathcal{F}_{\ell-\text{pict}}^{n,i}$  once for each threshold  $i \in [t]$  and then accumulate the result. We also use  $\mathcal{F}_{\ell-\text{geq}}^{n-t,v}$  functionality which is described in Sect. 2.  $\mathcal{F}_{\ell-\text{geq}}^{n-t,v}$  checks whether the secret shared values obtained from  $\mathcal{F}_{\ell-\text{pict}}^{n,i}$  indicates that the size of the intersection is greater than  $n - t$ . If that is the case  $\mathcal{F}_{\ell-\text{geq}}$  returns the exact size of the intersection, otherwise it returns the default value  $v$ . The protocol  $\Pi_{\ell-\cap}^{n,t,v}$  is described in Fig. 12.

**Theorem 8.** *Let  $n, t \in \mathbb{N}$  with  $n > t$  and  $v \in \mathbb{F}$ . The protocol  $\Pi_{\ell-\cap}^{n,t,v}$  depicted in Fig. 12 securely implements  $\mathcal{F}_{\ell-\cap}^{n,t,v}$  using one call to  $\mathcal{F}_{\ell-\text{pict}}^{n,i}$  for each  $i \in [t]$  and one call to  $\mathcal{F}_{\ell-\text{geq}}^{n-t,v}$ .*

*Proof.* For correctness, we observe that  $\mathcal{F}_{\ell-\text{pict}}^{n,i}(X_1, \dots, X_\ell)$  outputs sharing of 1, when the size of the intersection is greater than equal to  $n - i$ . Thus, if  $\left| \bigcap_{j=1}^\ell X_j \right| \geq n - t$  then  $\mathcal{F}_{\ell-\text{pict}}$  will return sharing of 1 exactly  $t - t^* + 1$  times, where  $n - t^*$  is the true intersection size. Consequently the protocol produces the correct output.

**Construction  $\Pi_{\ell-\cap}^{n,t,v}(X_1, \dots, X_\ell)$**

---

- 1: **for**  $i \in [t]$
- 2:      $(r_{1,i}, \dots, r_{\ell,i}) \leftarrow \mathcal{F}_{\ell-\text{pict}}^{n,i}(X_1, \dots, X_\ell)$
- 3: **for**  $i \in [\ell]$
- 4:     **Party**  $i$  :  $r_i := \sum_{j=1}^t r_{i,j}$
- 5: **Party** 1 :  $r_1 := n - t - 1 + r_1$
- 6:  $(d_1, \dots, d_\ell) \leftarrow \mathcal{F}_{\ell-\text{geq}}^{n-t,v}(r_1, \dots, r_\ell)$
- 7: **return**  $(d_1, \dots, d_\ell)$

**Fig. 12.** Protocol  $\Pi_{\ell-\cap}^{n,t,v}$  realizing  $\mathcal{F}_{\ell-\cap}^{n,t,v}$ .

*Privacy.* Without loss of generality assume that  $P_1, \dots, P_{\ell-1}$  are corrupted. The view of the corrupted parties only contain received messages from the oracles. Each oracle query to  $\mathcal{F}_{\ell\text{-pict}}^{n,t}$  returns a fresh secret sharing, which can be simulated by providing the corrupted parties with fresh shares of secret sharings of 0. The last query to  $\mathcal{F}_{\ell\text{-geq}}^{n-t,v}(r_1, \dots, r_\ell)$  can be simulated by returning the outputs given to the simulator. Indistinguishability of the simulated transcript from the real one directly follows from the security guarantees of additive secret sharing. □

We can use the protocol of Branco, Döttling, and Pu [BDP21] to instantiate  $\mathcal{F}_{\ell\text{-pict}}^{n,t}$  in  $\mathcal{F}_{\ell\text{-}\cap}^{n,t,v}$ . They present an  $\ell$ -party protocol with a communication complexity of  $\mathcal{O}(\lambda \ell^2 \text{polylog } t)$  bits based on additively homomorphic encryption. Their protocol can easily be extended to use generic secure computation techniques in all places, where additively homomorphic encryption was used. With this change, their protocol provides a solution based on, for instance, oblivious transfer with a communication complexity of  $\mathcal{O}(\lambda \ell \text{poly}(t))$  bits.

In our instantiation, we have  $\epsilon t$  buckets and for each of them we execute the protocol of Branco et al.  $\mathcal{O}(\ln t)$  times with a threshold of  $\mathcal{O}(\ln t)$ . Thus we get a total communication complexity of  $\mathcal{O}(\epsilon \lambda \ell t \text{polylog } t)$ .

**Corollary 5.** *Assuming the existence of oblivious transfer and or additively homomorphic encryption, there exists a protocol for securely computing the  $\ell$ -party private intersection cardinality test for threshold  $t$  with communication complexity of  $\mathcal{O}(\epsilon^2 \ell t \text{polylog } t)$  bits.*

Combining the results in our paper with the protocols for actually computing the intersection, once it is known that it is large enough, from by Ghosh and Simkin [GS19], we get the following result.

**Corollary 6.** *Assuming the existence of oblivious transfer or additively homomorphic encryption, there exists a passively secure protocol for threshold private set intersection among  $\ell$  parties with threshold  $t$  with communication complexity of  $\mathcal{O}(\epsilon^2 \ell t \text{polylog } t)$  bits.*

## References

[BDP21] Branco, P., Döttling, N., Pu, S.: Multiparty cardinality testing for threshold private intersection. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 32–60. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-75248-4\\_2](https://doi.org/10.1007/978-3-030-75248-4_2)

[BMRR21] Badrinarayanan, S., Miao, P., Raghuraman, S., Rindal, P.: Multi-party threshold private set intersection with sublinear communication. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 349–379. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-75248-4\\_13](https://doi.org/10.1007/978-3-030-75248-4_13)

[CD01] Cramer, R., Damgård, I.: Secure distributed linear algebra in a constant number of rounds. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 119–136. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44647-8\\_7](https://doi.org/10.1007/3-540-44647-8_7)

- [DCW13] Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: Sadeghi, A.-R., Gligor, V.D., Yung, M., (eds.) ACM CCS 2013: 20th Conference on Computer and Communications Security, Berlin, Germany, 4–8 November 2013, pp. 789–800. ACM Press (2013)
- [DPT20] Duong, T., Phan, D.H., Trieu, N.: Catalic: delegated psi cardinality with applications to contact tracing. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 870–899. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_29](https://doi.org/10.1007/978-3-030-64840-4_29)
- [FNP04] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24676-3\\_1](https://doi.org/10.1007/978-3-540-24676-3_1)
- [GS19] Ghosh, S., Simkin, M.: The communication complexity of threshold private set intersection. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 3–29. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26951-7\\_1](https://doi.org/10.1007/978-3-030-26951-7_1)
- [KKRT16] Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S., (eds.) ACM CCS 2016: 23rd Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016, pp. 818–829. ACM Press (2016)
- [KS05] Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005). [https://doi.org/10.1007/11535218\\_15](https://doi.org/10.1007/11535218_15)
- [Lin17] Lindell, Y.: How to simulate it—a tutorial on the simulation proof technique. In: *Tutorials on the Foundations of Cryptography*, pp. 277–346 (2017)
- [Mar14] Marlinspike, M.: The difficulty of private contact discovery (2014). <https://www.whispersystems.org/blog/contact-discovery>
- [Mea86] Meadows, C.A.: A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: *Proceedings of the 1986 IEEE Symposium on Security and Privacy*, Oakland, California, USA, 7–9 April, pp. 134–137 (1986)
- [NMH+10] Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., Borisov, N.: Botgrep: Finding P2P bots with structured graph analysis. In: *19th USENIX Security Symposium, Proceedings*, Washington, DC, USA, 11–13 August 2010, pp. 95–110 (2010)
- [PRTY19] Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: SpOT-light: lightweight private set intersection from sparse OT Extension. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 401–431. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_13](https://doi.org/10.1007/978-3-030-26954-8_13)
- [PRTY20] Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: PSI from PaXoS: fast, malicious private set intersection. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 739–767. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45724-2\\_25](https://doi.org/10.1007/978-3-030-45724-2_25)
- [PSSZ15] Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: Private set intersection using permutation-based hashing. In: *24th USENIX Security Symposium, USENIX Security 2015*, Washington, D.C., USA, 12–14 August 2015, pp. 515–530 (2015)

# **Encryption**





# Almost Tightly-Secure Re-randomizable and Replayable CCA-Secure Public Key Encryption

Antonio Faonio<sup>1</sup>, Dennis Hofheinz<sup>2</sup>, and Luigi Russo<sup>1</sup>(✉)

<sup>1</sup> EURECOM, Sophia Antipolis, France  
{faonio,russol}@eurecom.fr

<sup>2</sup> ETH Zurich, Zurich, Switzerland  
hofheinz@inf.ethz.ch

**Abstract.** Re-randomizable Replayable CCA-secure public key encryption (Rand-RCCA PKE) schemes guarantee security against chosen-ciphertext attacks while ensuring the useful property of re-randomizable ciphertexts. We introduce the notion of multi-user and multi-ciphertext Rand-RCCA PKE and we give the first construction of such a PKE scheme with an almost tight security reduction to a standard assumption. Our construction is structure preserving and can be instantiated over Type-1 pairing groups. Technically, our work borrows ideas from the state-of-the-art Rand-RCCA PKE scheme of Faonio *et al.* (ASIACRYPT'19) and the adaptive partitioning technique of Hofheinz (EUROCRYPT'17). Additionally, we show (1) how to turn our scheme into a publicly verifiable (pv) Rand-RCCA scheme and (2) that plugging our pv-Rand-RCCA PKE scheme into the MixNet protocol of Faonio *et al.* we can obtain the first *almost tightly-secure* MixNet protocol.

## 1 Introduction

Security against chosen-ciphertext attacks (CCA) is considered to be the standard notion of security for PKE schemes. This security definition, formulated by Rackoff and Simon [33], is elegant and easy to understand, and it has shown, by any means, to withstand the test of time.

**Replayable and Re-randomizable CCA Security.** Canetti, Krawczyk and Nielsen [7] pointed out that CCA security is not necessary for implementing secure channels. They showed that “replayable chosen-ciphertext” (RCCA) security suffices for secure channels, and might in fact allow for more efficient instantiations. Subsequently, Groth [20] showed that RCCA PKE schemes (called Rand-RCCA secure) can have re-randomizable ciphertexts. Specifically, Groth constructed a scheme with a ciphertext re-randomization procedure that, given a ciphertext as input, produces a fresh and unlinkable ciphertext that decrypts to the same message. Such a re-randomization procedure opens the door for applications that require secure communication *and* anonymity. For instance, Rand-RCCA secure PKE schemes enable anonymous and secure message transmissions (see Prabhakaran and Rosulek [32]), Mix-Nets (see Faonio *et al.* [13])

and Pereira and Rivest [31]), Controlled Functional Encryption (see Naveed *et al.* [30]), and one-round message-transmission protocols with reverse firewalls (see Dodis, Mironov and Stephens-Davidowitz [10]).

**Tight Security.** Yet another criticism of the original definition of CCA security is that while the definition postulates that the message underlying *one single* ciphertext remains protected even under CCA attacks, in the real world, a PKE scheme is used to protect a large number of ciphertexts from possibly many users. Now, it is well-known that security for one single ciphertext implies, through a hybrid argument, security for many ciphertexts and many users. However, it is unclear how much *concrete* security a PKE scheme really offers when it is used in the wild. This question, initially posed by Bellare, Boldyreva and Micali [4] created a fruitful area of research that investigates how tight the security of an encryption scheme translates to the trust that we have with respect to the cryptographic assumption that it relies on. In more detail, a tight security reduction ensures that for any attack on the PKE scheme, there exists an attack on the assumption that is similar both in terms of complexity (i.e. the running time, the space required, etc.) and success probability. Thus, in the setting of tight security reductions, the number of ciphertexts considered by the security definition matters.

By now, many CCA-PKE schemes have been proven to have tight security in the multi-ciphertext and multi-user setting: some notable examples are the works of [17, 18, 21, 22, 25, 26]. However, tight security in the context of Rand-RCCA security has not been studied.

## 1.1 Our Contributions

We initiate the study of tight security for Rand-RCCA secure PKE schemes in the multi-ciphertext and multi-user setting. Our main contributions are a new security definition for RCCA security in multi-ciphertext and multi-user setting (hereafter, mRCCA security), and a Rand-mRCCA PKE scheme whose mRCCA security (almost<sup>1</sup>) tightly reduces to the  $\mathcal{D}_d$ -MDDH assumption in symmetric (a.k.a. type-1) pairing groups.

Moreover, as an application, we revise the protocol for universally composable MixNet based on Rand-RCCA PKE from [13]. In the following paragraphs, we elaborate more about each of the contributions.

**Multi-user Multi-ciphertext RCCA Security.** In the security experiment of the (single-ciphertext) RCCA security notion, the decryption oracle, called “guarded decryption oracle”, can be queried on any ciphertext, including the challenge ciphertext. However, when decryption leads to one of the challenge messages  $(M_0, M_1)$ , the oracle answers with a special symbol  $\diamond$  (meaning “same”). As a warm-up, consider a trivial extension to the case of (single-user) multi-ciphertext RCCA security where the attacker is given:

- an encryption oracle that, on input a pair of messages  $M_0, M_1$ , returns some valid encryption of  $M_b$  where  $b$  is the challenge bit,

---

<sup>1</sup> As most of the tightly-secure schemes, the security reduction suffers from a small multiplicative loss that is however independent of the number of uses of the scheme.

- and a guarded decryption oracle that, on input a ciphertext  $C$ , returns a message  $M$ , or the special indexed symbol  $\diamond_j$  if  $C$  corresponds to an encryption of a message that was given as input to the encryption oracle as  $j$ -th query.

We notice that this trivial extension of RCCA security to multiple ciphertexts is impossible to achieve. Namely, consider the following generic attacker  $\mathcal{A}$  that makes three queries to the encryption oracle: (i)  $\mathcal{A}$  sends  $(M_1, M_2)$ , and receives back  $C_A$ ; (ii) sends  $(M_2, M_3)$ , and receives back  $C_B$ ; (iii) sends  $(M_3, M_1)$ , and receives back  $C_C$ .  $\mathcal{A}$  now queries the decryption oracle with  $C_C$ . If the bit  $b$  is 0, the decryption oracle returns  $\diamond_2$ ; if  $b$  is 1, the decryption oracle returns  $\diamond_1$ .

Yet another natural extension of the single-ciphertext RCCA security notion to the multi-ciphertext setting is to consider a guarded decryption oracle that upon input a ciphertext  $C$  either returns a message or the special symbol  $\diamond$ , but without notifying the adversary of which index  $j$  triggered the special symbol. Even if this definition avoids the attack described above, it is not as convenient as we would like it to be. Roughly speaking, the guarded decryption oracle reveals to the adversary that the queried ciphertext is a replay attack, but it doesn't tell which ciphertext was replayed; therefore, the larger the number of challenge ciphertexts, the less informative the output of the guarded decryption oracle will be. In particular, this definition is not sufficient for our MixNet application.

“In medio stat virtus”, as the saying goes: the definition we propose is weaker than the first attempted (yet impossible to achieve) definition, but stronger than the above-mentioned definition. To build some intuition, in an equivalent version of the single-ciphertext RCCA security definition, the guarded decryption oracle would output the minimal set of messages that the queried ciphertext could decrypt to and such that such set does not trivially break the RCCA security definition: namely, if the ciphertext is a replay attack then the oracle replies with the set of challenge messages  $\{M_0, M_1\}$ , otherwise with a message  $M' \notin \{M_0, M_1\}$ . We take a similar approach in our (multi-user) multi-ciphertext RCCA definition. The guarded decryption oracle outputs the minimal set of messages that the ciphertext could decrypt to without trivially breaking security. This set of messages includes all the pairs of challenge messages for which at least one of them is equal to the decryption of the queried ciphertext. To support the claim that our definition is indeed the most natural extension of RCCA to the multi-ciphertext setting, we prove that the simulation-based notion for RCCA security from [7] is tightly implied by our mRCCA security notion.

**A Tightly-Secure Rand-mRCCA PKE Scheme.** Our starting points are the recent work of Faonio *et al.* [13] (hereafter FFHR19), which is the state of art for Rand-RCCA PKE scheme, and the tightly-secure CCA PKE schemes based on the adaptive partitioning techniques of Hofheinz [22] and Gay *et al.* [19].

Very briefly, the main idea of our construction is to encrypt the message similarly to FFHR19, and append a non-interactive proof of consistency for (part of) the ciphertext; the latter proof needs to have a (weak) form of simulation soundness property that can be obtained information-theoretically. Namely, using the notation of [22], we append to the ciphertext a *benign proof* for the consistency of part of the ciphertext (which lies in a linear language) of a proof system that

is statistically sound even when the adversary has oracle access to simulated proofs for a larger language that includes the disjunction of two linear spaces.

**Some Technical Details.** To go from the rough idea described above to the actual scheme, we need to overcome two technical problems. The first problem is that our benign proof system needs to be re-randomizable (or, to better say, “malleable” as it needs to be able to re-randomize proofs of re-randomized statements), as we are aiming to construct a Rand-PKE scheme. We notice that none of the benign proof systems or affine notions we are aware of (such as [2, 18, 19, 22]) are re-randomizable. To solve this problem, we introduce a new malleable proof system based on the work of Abdalla, Benhamouda and Pointcheval [1].

The second (and more challenging) technical problem is that we need to reconcile the adaptive partitioning technique with the Rand-RCCA technique of [13]. In particular, at the core of the adaptive partitioning technique there is a complex argument that shows that the decryption oracle can safely reject *ill-formed* ciphertexts even when the adversary can observe (many) ill-formed challenge ciphertexts. In some sense, these challenge ciphertexts are the only ill-formed ciphertexts that correctly decrypt, while all other ill-formed ciphertexts produced by the adversary do not. However, in our security proof the adversary can easily produce ill-formed ciphertexts that correctly decrypt, simply by re-randomizing challenge ciphertexts.

In more detail, the adaptive partitioning technique moves the challenge ciphertexts back and forth between two different linear spaces (different from the linear space of honestly-generated ciphertexts). In our proof, differently than in previous works, we need to carefully define the relationship between these different linear spaces. In particular, it is necessary to make sure that re-randomizations of the challenge ciphertexts still lie in the prescribed linear space (and thus can be identified by our technique when answering  $\diamond$ ). More technically, a ciphertext for our scheme can be parsed as a vector  $[\mathbf{x}]$  in the source group (the CPA part of the ciphertext) plus two zero-knowledge proofs of consistency. The vector  $[\mathbf{x}]$  for a well-formed ciphertext lies in the affine space defined by the encrypted message and the span of a matrix  $[\mathbf{D}^*]$  which is part of the public key. Re-randomization works by summing up a random vector from the span of  $\mathbf{D}^*$  to  $\mathbf{x}$  (and updating the proofs accordingly). To apply the adaptive partitioning techniques, we move the challenge ciphertexts back and forth from two well-crafted distinct superspaces of  $\mathbf{D}^*$ . Thanks to this choice, we can recognize the challenge ciphertexts after re-randomization by multiplying the decrypted ciphertext by a matrix orthogonal to  $\mathbf{D}^*$ : this operation could be roughly interpreted as an “extended decryption” of the ciphertexts (since  $\mathbf{D}^*$  encodes partial information of the secret key), however, we are not only interested to identify the encrypted message but also to uniquely link the decrypted (possibly re-randomized) ciphertext with one of the challenge ciphertexts. Thus, like previous adaptive partitioning approaches, we separate the randomness space of the PKE scheme into an honest part (the span of  $\mathbf{D}^*$ ) and a normally unused part (spanned by the vectors in the mentioned super spaces, independent of  $\mathbf{D}^*$ ) that

is also used to hide the messages. In our view, the main technical insight is that the span of  $\mathbf{D}^*$  is used for re-randomization, while the other space is kept fixed for the challenge ciphertexts. We highlight that in order for the aforementioned strategy to work smoothly, we preferred to follow a flavor of adaptive partitioning as in Gay *et al.* [19], where secret keys are randomized, instead of the original strategy of Hofheinz [22], where ciphertexts are randomized. Finally, the original adaptive partitioning strategy relies on the pairwise universality of a hash proof system [9] that guarantees simpler statements about linear languages. We adapt this proof system to re-randomizable statements by considering higher-dimensional languages and refining the “core lemma for Rand-RCCA” from [13]. We highlight that this lemma was designed for the single-ciphertext scenario, thus, some extra care is needed in our adaptive partitioning argument, more in detail, when defining the notion of *critical query*. In particular, a critical query is commonly defined as a decryption query for an ill-formed ciphertext that would decrypt without errors under one of the randomized secret keys; the usual goal is to show that an adversary cannot make such a query. In our case, we need to refine this notion by additionally specifying when (allegedly) re-randomizations of challenge ciphertexts are critical. Since each one of the challenge ciphertexts is an ill-formed ciphertext that decrypts correctly under one of the randomized keys, we cannot consider critical a re-randomization of such a challenge ciphertext when it decrypts correctly under the same randomized key. Thus, after having recognized a decryption query as a re-randomization, we make sure that this ciphertext is decrypted only using a specific (a univocally linked) secret key; on the other hand, other kinds of decryption queries can be safely decrypted with any of the secret keys. This rule allows eventually to use the lemma of [13] which provides security even given an interface for decryption of re-randomizations of one challenge ciphertext under one specific secret key.

**Extensions and Applications.** Following the strategy of [13] we show that our Rand-mRCCA PKE can be used to instantiate a PKE with the nice property of publicly verifiable ciphertexts (pv-Rand-mRCCA PKE). We propose two pv-Rand-mRCCA PKE schemes: one based on the Matrix Diffie-Hellman Assumption (MDDH), and a second more efficient scheme based on a new MDDH-like assumption (see Sect. 1.2 for the details) which we prove secure in the generic group model.

As an application of our framework, we show that we can plug a pv-Rand-mRCCA scheme into the MixNet protocol of [13]. Instantiating such protocol with our schemes, we obtain an (almost) “*tightly-secure*” MixNet protocol: namely a protocol, the first of its kind, whose security guarantees depend linearly on the number of mixer parties but only logarithmically on the number of mixed messages. To compare with the state of the art for MixNet protocols, we notice that the Bayer and Groth [3] proof of shuffle is based on the Fiat-Shamir transform applied to a multi-round Sigma protocol, thus the security reduction degrades with the number of rounds of the underlying Sigma protocol, while the proof of shuffle in the pairing setting of Fauzi *et al.* [16] relies on new kinds of  $\mathcal{D}_n$ -KerMDH assumptions (proved to hold generically in the same paper) where  $n$  is the number of shuffled ciphertexts.

## 1.2 Related Work

Prabhakaran and Rosulek [32] introduced the first Rand-RCCA PKE in the standard model. Abstracting the scheme of [32], and solving a long-standing open problem, recently Wang *et al.* [35] introduced the first receiver-anonymous Rand-RCCA PKE. Faonio and Fiore [12] introduced a practical Rand-RCCA PKE in the random oracle model. Considering the state of the art on pairing-based Rand-RCCA PKE schemes, the most relevant works are the Rand-RCCA PKE scheme of Chase *et al.* [8], the recent works of Libert, Peters and Qian [27], and of Faonio *et al.* [13]. In Table 1 we offer a comparison, in terms of security properties and functionalities, of our schemes of Sect. 5, i.e.  $\mathcal{PK}\mathcal{E}_1$ ,  $\mathcal{PK}\mathcal{E}_2$  and  $\mathcal{PK}\mathcal{E}_3$ , and the previous schemes. From a technical point of view, our schemes inherit from the scheme of [13], however, we notice that our schemes are instantiated on type-1 pairing group, while FFHR19 is instantiated on type-3 pairing group (see the next section and [14] for more details). On the other hand, our schemes are the only ones that have (almost) tight-security reductions. In Table 2 we compare the most efficient Rand-RCCA PKE schemes with ours. In particular, we instantiate  $\mathcal{PK}\mathcal{E}_1$  and  $\mathcal{PK}\mathcal{E}_2$  under DLIN assumption for type-1 pairing group ( $d = 2$  and, because of the security of the benign proof system,  $n = 6$ ) while we instantiate  $\mathcal{PK}\mathcal{E}_3$  under  $\mathcal{U}_{9,4}$ -TMDDH assumption. We compare the number of operations required by the three algorithms (Enc, Rand and Dec) and the size of the ciphertext. In particular, we have considered the cost of exponentiations in the source and target groups, and the number of pairings. We give only a rough estimation of the costs of  $\mathcal{PK}\mathcal{E}_2$  and  $\mathcal{PK}\mathcal{E}_3$  to provide some intuition on the considerable efficiency gap between them: their cost is derived in terms of group elements and operations needed to instantiate the proof systems for  $\mathcal{PK}\mathcal{E}_2$  (resp.  $\mathcal{PK}\mathcal{E}_3$ ) under  $\mathcal{D}_{6,2}$ -MDDH (resp.  $\mathcal{U}_{9,4}$ -TMDDH) assumption from [11] and [13].

We note that  $\mathcal{PK}\mathcal{E}_2$  and  $\mathcal{PK}\mathcal{E}_3$  are far from being considered practical, while  $\mathcal{PK}\mathcal{E}_1$  is considerably less efficient than [13]. Indeed, our main goal is to prove feasibility. We view our work as a potential first towards a tightly secure practical solution. For instance, while the first tightly IND-CCA secure PKE schemes were highly impractical, state-of-the-art schemes (see [17, 18]) have a realistic break-even point<sup>2</sup>. We hope for a similar development with Rand-RCCA PKE schemes.

Our benign proof system uses the “OR-Proof” technique from [1]. We notice that, in the context of tightly-secure reductions, the same technique from [1] has been used in [21] to instantiate their (Leakage-Resilient) Ardent Quasi-Adaptive Hash Proof System. We stress that in our work, in contrast with [21], the main reason to use the technique from [1] is because of its nice linear property that, in turn, allows for *malleable* proof system.

## 1.3 Open Problems

Our Rand-RCCA PKE schemes require type-1 pairing groups, which are less efficient than type-3. It is natural to ask whether we can instantiate our PKE

<sup>2</sup> For the same security parameter, the work of [17, 18] outperforms state-of-the-art non-tightly secure schemes like Kurosawa-Desmedt [24] around  $2^{30}$  ciphertexts.

**Table 1.** Comparison of the properties of a selection of Rand-RCCA-secure PKE schemes. The symbol \* indicates that the structure-preserving property of the schemes is not strict since ciphertexts contain some elements in  $\mathbb{G}_T$ .

PKE	Group Setting	Assumption	Struc. Pres.	Pub. Ver.	Tight
[8] CKLM12, [27] LPQ17	Type-3	SXDH	✓	✓	
[13] FFHR19	Type-3	$\mathcal{D}_{d+1,d}$ -MDDH	✓*	✓	
$\mathcal{PK}\mathcal{E}_1$	Type-1	$\mathcal{D}_{n,d}$ -MDDH	✓*		✓
$\mathcal{PK}\mathcal{E}_2$	Type-1	$\mathcal{D}_{n,d}$ -MDDH	✓*	✓	✓
$\mathcal{PK}\mathcal{E}_3$	Type-1	$\mathcal{U}_{n,d}$ -TMDDH	✓*	✓	✓

**Table 2.** Efficiency comparison among the best Rand-RCCA-secure PKE schemes. We denote as  $E_i$  the cost of 1 exponentiation in  $\mathbb{G}_i$ ,  $P$  the cost of computing a bilinear pairing. In the third column, we consider the cost of **Enc** which is almost always comparable with the cost of **Rand**. The first two schemes are privately verifiable, while the last four are publicly verifiable. We consider the most efficient instantiations for  $\mathcal{PK}\mathcal{E}_1, \mathcal{PK}\mathcal{E}_2$  (DLIN), for  $\mathcal{PK}\mathcal{E}_3$  ( $\mathcal{U}_{9,4}$ -TMDDH) and for [13] (SXDH).

PKE	C	Enc $\approx$ Rand	Dec
[13] FFHR19 (1)	$3G_1 + 2G_2 + G_T$	$4E_1 + 5E_2 + 2E_T + 5P$	$8E_1 + 4E_2 + 4P$
$\mathcal{PK}\mathcal{E}_1$	$7G_1 + 2G_T$	$14E_1 + 2E_T + 14P$	$48E_1 + 36E_T + 49P$
[27] LPQ17	$42G_1 + 20G_2$	$79E_1 + 64E_2$	$1E_1 + 142P$
[13] FFHR19 (2)	$14G_1 + 15G_2 + 4G_T$	$36E_1 + 45E_2 + 6E_T + 5P$	$2E_1 + 50P$
$\mathcal{PK}\mathcal{E}_2$	$380G_1 + 330G_T$	$\approx 180E_1 + 110E_T + 38P$	$\approx 6E_1 + 400P$
$\mathcal{PK}\mathcal{E}_3$	$105G_1 + 9G_T$	$\approx 261E_1 + 9E_T + 16P$	$\approx 6E_1 + 11P$

schemes from type-3 pairings. Unfortunately, we do not know how to do so, because it is not clear how to reconcile the adaptive partitioning technique [22] with a Rand-RCCA construction in settings with type-3 pairings (such as the one from [13]). We elaborate more on the challenges to overcome for obtaining a type-3 instantiation in [14] and leave the construction of a tightly-secure type-3 Rand-RCCA PKE scheme as an interesting open problem.

Our approach is semi-generic, as we work with pairing-based cryptography. We leave as open problem to provide a generic framework to instantiate (almost) tightly-secure Rand-RCCA-secure PKE. Possible starting points are the HPS-based frameworks of [35] for Rand-RCCA schemes and [21] for tightly-secure (LR-)CCA-secure schemes. Recently, Faonio and Russo [15] improved over the mix-net protocol of [13], giving a more efficient instantiation based on non publicly-verifiable Rand-RCCA PKE schemes; however, their construction requires a leakage-resilient scheme. We leave as open problem the extension of our analysis to tightly-secure LR-RCCA PKE schemes to extend their approach.

## 2 Preliminaries

A function is negligible in  $\lambda$  if it vanishes faster than the inverse of any polynomial in  $\lambda$ . We write  $f(\lambda) \in \text{negl}(\lambda)$  when  $f$  is negligible in  $\lambda$ . For any bit string  $\tau \in \{0, 1\}^*$ , we denote by  $\tau[i]$  the  $i$ -th bit of  $\tau$  and by  $\tau_i$  the bit string comprising the

first  $i$  bits of  $\tau$ . A symmetric (type-1) bilinear group  $\mathcal{G}$  is a tuple  $(q, \mathbb{G}_1, \mathbb{G}_T, e, \mathcal{P}_1)$ , where  $\mathbb{G}_1$  and  $\mathbb{G}_T$  are groups of prime order  $q$ , the element  $\mathcal{P}_1$  is a generator of  $\mathbb{G}_1$ ,  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$  is an efficiently computable, non-degenerate bilinear map. Let  $\text{GGen}$  be a PPT algorithm which on input  $1^\lambda$ , where  $\lambda$  is the security parameter, returns a description of a symmetric bilinear group  $\mathcal{G}$ . Elements in  $\mathbb{G}_i$ , are denoted in implicit notation as  $[a]_i := a\mathcal{P}_i$ , where  $i \in \{1, T\}$  and  $\mathcal{P}_T := e(\mathcal{P}_1, \mathcal{P}_1)$ . Every element in  $\mathbb{G}_i$  can be written as  $[a]_i$  for some  $a \in \mathbb{Z}_q$ , but note that given  $[a]_i$ ,  $a \in \mathbb{Z}_q$  is in general hard to compute (discrete logarithm problem). Given  $a, b \in \mathbb{Z}_q$  we distinguish between  $[ab]_i$ , namely the group element whose discrete logarithm base  $\mathcal{P}_i$  is  $ab$ , and  $[a]_i \cdot b$ , namely the execution of the multiplication of  $[a]_i$  and  $b$ , and  $[a]_1 \cdot [b]_1 = [a \cdot b]_T$ , namely the execution of a pairing between  $[a]_1$  and  $[b]_1$ . Sometimes, to simplify the notation, we will write  $[a]$  instead of  $[a]_1$  for elements in the source group. Vectors and matrices are denoted in boldface. We extend the pairing operation to vectors and matrices as  $e([\mathbf{A}]_1, [\mathbf{B}]_1) = [\mathbf{A}^\top \cdot \mathbf{B}]_T$  and  $e([y]_1, [\mathbf{A}]_1) = [y \cdot \mathbf{A}]_T$ . Let  $\text{span}(\mathbf{A})$  denote the linear span of the columns of  $\mathbf{A}$ .  $\mathcal{D}_{n,d}$  is a matrix distribution if outputs (in probabilistic polynomial time, with overwhelming probability) matrices in  $\mathbb{Z}_q^{n \times d}$ .

**Definition 1 (Matrix Decisional Diffie-Hellman Assumption, [11]).** *The  $\mathcal{D}_{n,d}$ -MDDH assumption holds if for all non-uniform PPT adversaries  $\mathcal{A}$ ,*

$$|\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{A}\mathbf{w}]_1) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{z}]_1) = 1]| \in \text{negl}(\lambda),$$

where the probability is taken over  $\mathcal{G} = (q, \mathbb{G}_1, \mathbb{G}_T, e, \mathcal{P}_1) \leftarrow \text{GGen}(1^\lambda)$ ,  $\mathbf{A} \leftarrow \mathcal{D}_{n,d}$ ,  $\mathbf{w} \leftarrow \mathbb{Z}_q^d$ ,  $[\mathbf{z}]_1 \leftarrow \mathbb{G}_1^n$  and the coin tosses of adversary  $\mathcal{A}$ .

For  $Q \in \mathbb{N}$ ,  $\mathbf{W} \leftarrow_{\$} \mathbb{Z}_q^{d \times Q}$  and  $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{n \times Q}$ , the  $Q$ -fold  $\mathcal{D}_{n,d}$ -MDDH assumption states that distinguishing tuples of the form  $([\mathbf{A}]_1, [\mathbf{A}\mathbf{W}]_1)$  from  $([\mathbf{A}]_1, [\mathbf{U}]_1)$  is hard. That is, a challenge for the  $Q$ -fold  $\mathcal{D}_{n,d}$ -MDDH assumption consists of  $Q$  independent challenges of the  $\mathcal{D}_{n,d}$ -MDDH Assumption (with the same  $\mathbf{A}$  but different randomness  $\mathbf{w}$ ). In [11] it is shown that the two problems are equivalent, where the reduction loses at most a factor  $n - d$ .

**Tensor Product.** Let  $\mathbf{a} \in \mathbb{Z}_q^n$  and  $\mathbf{b} \in \mathbb{Z}_q^{n'}$ , we define  $\mathbf{a} \otimes \mathbf{b} \in \mathbb{Z}_q^{nn'}$  to be the tensor product between the two vectors. We can show the following property:

$$(\mathbf{A} \cdot \mathbf{R}) \otimes (\mathbf{B} \cdot \mathbf{S}) = (\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{R} \otimes \mathbf{S}) \tag{1}$$

**Lemma for Rand-RCCA Security.** The main technical tool employed by [13], to which they refer as their “core lemma”, roughly speaking says that, for any  $\mathbf{u} \in \mathbb{Z}_q^{d+1}$ , the projective hash function with hash key  $\mathbf{f}, \mathbf{F}$  that maps  $\mathbf{v}$  to  $(\mathbf{f} + \mathbf{F}\mathbf{v})^\top \mathbf{u}$  is pair-wise independent with respect to the quotient set  $\mathbb{Z}_q^{d+2} / \text{span}(\mathbf{E})$  when given as side information the matrix  $\mathbf{F}\mathbf{E}$  where  $\mathbf{E} \in \mathbb{Z}_q^{d+2 \times d}$ . We generalize their result to  $\mathbf{u} \in \mathbb{Z}_q^n$  and  $\mathbf{E} \in \mathbb{Z}_q^{n' \times d}$  for any  $n > d$  and  $n' > d + 1$ . The proof of the lemma follows by reduction to the original lemma from [13] and it can be found in [14]. For the sake of clarity, in this paper we prefer to call this lemma the “Rand-RCCA lemma”, rather than “core lemma” (for Rand-RCCA) as in [13], because the core technical parts of our work and theirs are different.



**Lemma 1 (Rand-RCCA Lemma).** *Let  $d$  be a positive integer. For any matrix  $\mathbf{D} \in \mathbb{Z}_q^{n \times d}$ ,  $\mathbf{E} \in \mathbb{Z}_q^{n' \times d}$  where  $n > d$  and  $n' > d + 1$ , and any (possibly unbounded) adversary  $A$ :*

$$\Pr \left[ \begin{array}{l} \mathbf{u} \notin \text{span}(\mathbf{D}) \\ (\mathbf{v} - \mathbf{v}^*) \notin \text{span}(\mathbf{E}) : (z, \mathbf{u}, \mathbf{v}) \leftarrow_{\$} \mathcal{A}^{\mathcal{O}}(\mathbf{D}, \mathbf{E}, \mathbf{f}^\top \mathbf{D}, \mathbf{F}^\top \mathbf{D}, \mathbf{F}\mathbf{E}) \\ z = (\mathbf{f} + \mathbf{F}\mathbf{v})^\top \mathbf{u} \end{array} \right] \leq \frac{n \cdot n'}{q}.$$

where the adversary outputs a single query  $\mathbf{v}^*$  to  $\mathcal{O}$  that returns  $\mathbf{f} + \mathbf{F} \cdot \mathbf{v}^*$ .

### 3 Non-Interactive Proof Systems (NIPS)

**Definition 2 (Proof system).** *Let  $\mathcal{L} = \{\mathcal{L}_{\text{pars}}\}$  be a family of languages with  $\mathcal{L}_{\text{pars}} \subseteq \mathcal{X}_{\text{pars}}$ , and with efficiently computable witness relation  $\mathcal{R}$ . A non-interactive proof system (NIPS)  $\mathbf{PS} = (\text{PGen}, \text{PPrv}, \text{PVer}, \text{PSim})$  for  $\mathcal{L}$  consists of the following PPT algorithms:*

- $\text{PGen}(1^\lambda, \text{pars})$  outputs a proving key  $\text{ppk}$ , a verification key  $\text{psk}$ .
- $\text{PPrv}(\text{ppk}, x, w)$ ,  $x \in \mathcal{L}$  and  $\mathcal{R}(x, w) = 1$ , outputs a proof  $\pi$ .
- $\text{PVer}(\text{psk}, x, \pi)$ ,  $x \in \mathcal{X}$  and a proof  $\pi$ , outputs a verdict  $b \in \{0, 1\}$ .
- $\text{PSim}(\text{psk}, x)$ ,  $x \in \mathcal{L}$ , outputs a proof  $\pi$ .

**Completeness:** *For all  $\text{pars}$ , all  $(\text{ppk}, \text{psk})$  in the range of  $\text{PGen}(1^\lambda, \text{pars})$ , all  $x \in \mathcal{L}$ , and all  $w$  with  $\mathcal{R}(\text{pars}, x, w) = 1$ , we have  $\text{PVer}(\text{psk}, x, \text{PPrv}(\text{ppk}, x, w)) = 1$ .*

When  $\text{ppk} \neq \text{psk}$  we say that the proof system is *designated verifier*. In the definition above we let the verification and proving key depend on the parameters of the relation, namely, the proof systems are *quasi-adaptive* as defined by Jutla and Roy [23]. All the NIPs of this paper are *structure-preserving*: i.e., all the public interfaces are vectors in the source groups, all the private material is in  $\mathbb{Z}_q$  and all the algorithms can be described with pairing-product equations; also, as in [13] the proof  $\pi$  could lie in the target group.

**Benign Proof Systems.** All relevant security properties of a benign NIDVPS are condensed in the following definitions, taken verbatim from [22].

**Definition 3 (Benign proof system).** *Let  $\mathbf{PS}$  be an NIDVPS for  $\mathcal{L}$  as in Definition 2, and let  $\mathcal{L}^{\text{sim}} = \{\mathcal{L}_{\text{pars}}^{\text{sim}}\}$ ,  $\mathcal{L}^{\text{ver}} = \{\mathcal{L}_{\text{pars}}^{\text{ver}}\}$ , and  $\mathcal{L}^{\text{snd}} = \{\mathcal{L}_{\text{pars}}^{\text{snd}}\}$  be families of languages. We say that  $\mathbf{PS}$  is  $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$ -benign if the following properties hold:*

**(Perfect) zero-knowledge.** *For all  $\text{pars}$ , all  $(\text{ppk}, \text{psk})$  that lie in the range of  $\text{PGen}(1^\lambda, \text{pars})$ , and all  $x \in \mathcal{L}$  and  $w$  with  $\mathcal{R}(\text{pars}, x, w) = 1$ , we have that the distribution  $\text{PPrv}(\text{ppk}, x, w)$  is equivalent to  $\text{PSim}(\text{psk}, x)$ .*

**(Statistical)  $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$ -soundness.** *Let  $\text{Exp}_{\mathbf{A}, \mathbf{PS}}^{\text{snd}}$  be the game played by  $\mathbf{A}$  in Fig. 1. Let  $\text{Adv}_{\mathbf{PS}, \mathbf{A}}^{\text{snd}}(\lambda)$  be the probability that  $\text{Exp}_{\mathbf{A}, \mathbf{PS}}^{\text{snd}}(\lambda) = 1$ . We require that for all (possibly unbounded)  $\mathbf{A}$  that only make a polynomial number of oracle queries,  $\text{Adv}_{\mathbf{PS}, \mathbf{A}}^{\text{snd}}(\lambda)$  is negligible.*

**Non-Interactive Zero-Knowledge Proof Systems.** We adapt Definition 2 for the case of publicly verifiable proof systems by requiring the prover key and the verification key to be identical, and we refer to such key as the *common reference string*. (Nontrivial) proof systems with this syntax are commonly called zero-knowledge proof systems (NIZKs).

Notice that in the syntax of proof system we give in Definition 3 both the simulator  $\text{PSim}$  and the verifier  $\text{PVer}$  receive as input the verification key, while in the usual definition of NIZK the simulator receives a simulation trapdoor. This difference is only syntactical. We say that a NIZK  $\mathbf{PS}$  for  $\mathcal{L}$  is *adaptively sound* if it is statistically  $(\emptyset, \mathcal{L}, \emptyset)$ -sound according to Definition 3.

**Definition 4.** Let  $\mathbf{PS}$  be a NIPS for  $\mathcal{L}$  as in Definition 2, we say that  $\mathbf{PS}$  is  $(\epsilon, T)$ -composable zero-knowledge if there exists a PPT algorithm  $\text{PGen}$  such that:

- For all  $\text{pars}$ , the distributions induced by the first output of  $\text{PGen}(1^\lambda, \text{pars})$  and  $\overline{\text{PGen}}(1^\lambda, \text{pars})$  are  $\epsilon$ -close for any adversary with running time  $T$ .
- For all  $\text{pars}$ , all  $(\text{ppk}, \text{psk})$  that lie in the range of  $\overline{\text{PGen}}(1^\lambda, \text{pars})$ , and all  $x \in \mathcal{L}$  and  $w$  with  $\mathcal{R}(\text{pars}, x, w) = 1$ , we have that the distribution  $\text{PPrv}(\text{ppk}, x, w)$  is equivalent to  $\text{PSim}(\text{psk}, x)$ .

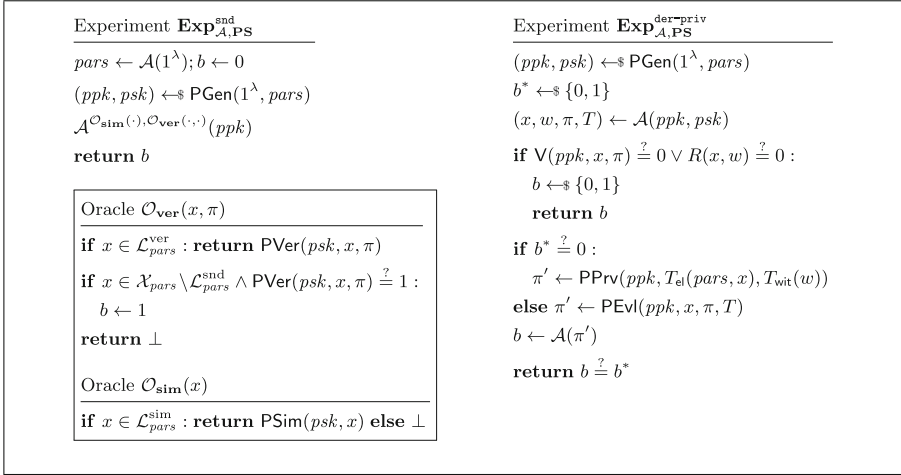
**Malleable NIPS.** We use the definitional framework of Chase *et al.* [8] for malleable proof systems. For simplicity of the exposition we consider only the unary case for transformations (see the aforementioned paper for more details). Moreover, we adapt their definition to the quasi-adaptive setting by having a transformation that depends on the  $\text{pars}$ . Let  $T = (T_{\text{el}}, T_{\text{wit}})$  be a pair of efficiently computable functions, that we refer to as a *transformation*.

**Definition 5 (Admissible transformation).** We say that an efficient relation  $\mathcal{R}$  is closed under a transformation  $T = (T_{\text{el}}, T_{\text{wit}})$  if for any  $(\text{pars}, x, w) \in \mathcal{R}$  the pair  $(\text{pars}, T_{\text{el}}(\text{pars}, x), T_{\text{wit}}(w)) \in \mathcal{R}$ . If  $\mathcal{R}$  is closed under  $T$  then we say that  $T$  is admissible for  $\mathcal{R}$ . Let  $\mathcal{T}$  be a set of transformations, if for every  $T \in \mathcal{T}$ ,  $T$  is admissible for  $\mathcal{R}$ , then  $\mathcal{T}$  is an allowable set of transformations.

**Definition 6 (Malleable NIPS).** Let  $\mathbf{PS}$  be an NIPS for  $\mathcal{L}$  as in Definition 2, and let  $\text{PEvl}(\text{ppk}, x, \pi, T)$  be a PPT algorithm that takes as inputs  $\text{ppk}$ , an instance  $x$ , a proof  $\pi$ , and a transformation  $T \in \mathcal{T}$ , and it outputs a proof  $\pi'$ . We say that  $\mathbf{PS}$  and  $\text{PEvl}$  form a malleable proof system for  $\mathcal{L}$  with set  $\mathcal{T}$  of allowable transformations for  $\mathcal{R}$ , if, for all  $\text{pars}$ ,  $(\text{ppk}, \text{psk})$  that lie in the range of  $\text{PGen}(1^\lambda, \text{pars})$ , all  $T \in \mathcal{T}$ , and all  $x, \pi$  we have  $\text{PVer}(\text{psk}, T_{\text{el}}(\text{pars}, x), \pi') = 1$  if and only if  $\text{PVer}(\text{psk}, x, \pi) = 1$ .

**Definition 7 (Derivation Privacy).** Let  $\mathbf{PS}$  be a malleable NIPS for  $\mathcal{L}$  with relation  $\mathcal{R}$  and an allowable set of transformations  $\mathcal{T}$  and corresponding  $\text{PEvl}$ . We say that  $\mathbf{PS}$  is derivation private if for any PPT adversary  $A$ :

$$\text{Adv}_{\mathcal{A}, \mathbf{PS}}^{\text{der-priv}}(\lambda) := \left| \Pr \left[ \text{Exp}_{\mathcal{A}, \mathbf{PS}}^{\text{der-priv}}(\lambda) = 1 \right] - \frac{1}{2} \right| \in \text{negl}(\lambda)$$



**Fig. 1.** Security experiments for benign soundness and derivation privacy of NIPS.

where  $\text{Exp}^{\text{der-priv}}$  is the game described in Fig. 1. Moreover we say that **PS** is perfectly (resp. statistically) derivation private when for any (possibly unbounded) adversary the advantage above is 0 (resp. negligible).

Similarly to [13], we require a technical property to show re-randomizability of our encryption scheme that we call *tightness for proofs*, which roughly speaking says that it is hard to find a proof for a valid instance that does not lie in the set of the proofs created by the prover. For space reasons, we give more details in [14].

### 3.1 Our Malleable NIDVPS Based on Type-1 Pairing

Let  $\mathbf{D} \in \mathbb{Z}_q^{n \times d}$ . We show that the following **PS** is a NIPS for  $\mathcal{L} = \text{span}([\mathbf{D}]_1)$ :

- $\text{PGen}(\text{pars})$  parses  $\text{pars}$  as  $\text{prm}_G, [\mathbf{D}]_1 \in \mathbb{G}_1^{n \times d}$  where  $n, d \in \mathbb{N}$ , samples  $\mathbf{k} \leftarrow \mathbb{Z}_q^{n^2}$ , let  $\mathbf{I}_n$  be the identity matrix of dimension  $n$ , set:

$$\text{psk} \leftarrow \mathbf{k} \text{ and } \text{ppk} \leftarrow (\mathbf{k}^\top [\mathbf{D} \otimes \mathbf{I}_n]_1, \mathbf{k}^\top [\mathbf{I}_n \otimes \mathbf{D}]_1, \mathbf{k}^\top [\mathbf{D} \otimes \mathbf{D}]_T)$$

- $\text{PPrv}(\text{ppk}, [\mathbf{u}]_1, \mathbf{r})$  computes  $\pi \leftarrow \mathbf{k}^\top [\mathbf{D} \otimes \mathbf{D}]_T \cdot (\mathbf{r} \otimes \mathbf{r})$  for  $[\mathbf{u}]_1 = [\mathbf{D}]_1 \mathbf{r}$
- $\text{PSim}(\text{psk}, [\mathbf{u}]_1)$  computes  $\pi \leftarrow \mathbf{k}^\top ([\mathbf{u}]_1 \otimes [\mathbf{u}]_1)$
- $\text{PVer}(\text{psk}, [\mathbf{u}]_1, \pi)$  returns 1 if and only if  $\mathbf{k}^\top ([\mathbf{u}]_1 \otimes [\mathbf{u}]_1) \stackrel{?}{=} \pi$

The first two vectors in the  $\text{ppk}$  are necessary to enable for the malleability of the proof system. While the third element of the public key could be efficiently derived from the previous two, we decide to publish it to speed up re-randomization and proving time. Consider the set  $\mathcal{T}$  of admissible transformations for  $\mathbb{Z}_q^n$ :

$$\mathcal{T} = \{T : T_{\text{el}}(\text{pars}, [\mathbf{u}]_1) = [\mathbf{u}]_1 + [\mathbf{D}]_1 \hat{\mathbf{r}}; T_{\text{wit}}(\mathbf{r}) = \mathbf{r} + \hat{\mathbf{r}}\} \tag{2}$$

We note that any transformation  $T$  in the set above is uniquely determined by the vector  $\hat{\mathbf{r}}$ , thus, whenever it is clear from the context, we will simply use  $\hat{\mathbf{r}}$  to identify the transformation. Let  $\text{PEVl}(ppk, \hat{\mathbf{r}}, [\mathbf{u}]_1, \pi)$  the algorithm that computes

$$\hat{\pi} \leftarrow \pi + \mathbf{k}^\top [\mathbf{I}_n \otimes \mathbf{D}]_1 \cdot [\mathbf{u} \otimes \hat{\mathbf{r}}]_1 + \mathbf{k}^\top [\mathbf{D} \otimes \mathbf{I}_n]_1 \cdot [\hat{\mathbf{r}} \otimes \mathbf{u}]_1 + \mathbf{k}^\top [\mathbf{D} \otimes \mathbf{D}]_T \cdot \hat{\mathbf{r}} \otimes \hat{\mathbf{r}}.$$

We show that  $\mathbf{PS}$  and  $\text{PEVl}$  form a malleable proof system for the set of transformation  $\mathcal{T}$  and the language  $\mathcal{L}$ .

**Theorem 1.** *Let  $\mathcal{L} = \text{span}([\mathbf{D}]_1)$  and let  $\mathcal{L}^{\text{snd}} = \mathcal{L}^{\text{sim}} = \{[\mathbf{u}]_1 : [\mathbf{u}]_1 = [\mathbf{D}_0]_1 \mathbf{r} \vee [\mathbf{u}]_1 = [\mathbf{D}_1]_1 \mathbf{r}\}$ , and  $\mathcal{L}^{\text{ver}} = \mathbb{Z}_q^n$ , where  $\mathbf{D}_i = \mathbf{D} \parallel \bar{\mathbf{D}}_i$  for  $i \in \{0, 1\}$ ,  $\mathbf{D} \in \mathbb{Z}_q^{n \times d}$  and  $\bar{\mathbf{D}}_0, \bar{\mathbf{D}}_1 \in \mathbb{Z}_q^{n \times d'}$ .  $\mathbf{PS}$  is a  $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$ -benign proof system for  $\mathcal{L}$  as long as  $n^2 > 2n \cdot d + 2d'^2$ , moreover,  $\mathbf{PS}$  and  $\text{PEVl}$  form a malleable proof system for  $\mathcal{L}$  and the set of transformation  $\mathcal{T}$  defined in Eq. (2).*

*Proof.* In what follows, we prove each of the properties.

**Completeness and Malleability.** Our benign proof system is complete, as by Eq. (1) for any  $\mathbf{u} = \mathbf{D}\mathbf{r}$  we have  $(\mathbf{u} \otimes \hat{\mathbf{r}}) = (\mathbf{D} \otimes \mathbf{D}) \cdot (\mathbf{r} \otimes \hat{\mathbf{r}})$ . We prove that our scheme is *malleable* (Definition 6) with respect to set of transformation  $\mathcal{T}$  defined in Eq. (2), i.e., we prove that for any  $[\mathbf{u}]$  and any  $\hat{\mathbf{r}}$ , a proof  $\pi$  for  $[\mathbf{u}]$  verifies if and only if the proof  $\hat{\pi}$  obtained executing  $\text{PEVl}$  on  $\pi$  and the transformation  $\hat{\mathbf{r}}$  verifies for  $[\mathbf{u} + \mathbf{D}\hat{\mathbf{r}}]$ . For the first direction of the implication:

$$\begin{aligned} \hat{\pi} &= \pi + \mathbf{k}^\top (\mathbf{I}_n \otimes \mathbf{D}) \cdot (\mathbf{u} \otimes \hat{\mathbf{r}}) + \mathbf{k}^\top (\mathbf{D} \otimes \mathbf{I}_n) \cdot (\hat{\mathbf{r}} \otimes \mathbf{u}) + \mathbf{k}^\top (\mathbf{D} \otimes \mathbf{D}) \cdot (\hat{\mathbf{r}} \otimes \hat{\mathbf{r}}) \\ &= \mathbf{k}^\top (\mathbf{u} \otimes \mathbf{u}) + \mathbf{k}^\top ((\mathbf{I}_n \mathbf{u}) \otimes (\mathbf{D}\hat{\mathbf{r}})) + \mathbf{k}^\top ((\mathbf{D}\hat{\mathbf{r}}) \otimes (\mathbf{I}_n \mathbf{u})) + \mathbf{k}^\top ((\mathbf{D}\hat{\mathbf{r}}) \otimes (\mathbf{D}\hat{\mathbf{r}})) \\ &= \mathbf{k}^\top (\mathbf{u} \otimes \mathbf{u} + \mathbf{u} \otimes (\mathbf{D}\hat{\mathbf{r}}) + (\mathbf{D}\hat{\mathbf{r}}) \otimes \mathbf{u} + (\mathbf{D}\hat{\mathbf{r}}) \otimes (\mathbf{D}\hat{\mathbf{r}})) \\ &= \mathbf{k}^\top ((\mathbf{u} + \mathbf{D}\hat{\mathbf{r}}) \otimes (\mathbf{u} + \mathbf{D}\hat{\mathbf{r}})) \end{aligned}$$

We highlight that the second equation holds because of the definition of  $\pi$  and (1), while the third equation is obtained by grouping the previous line by  $\mathbf{k}^\top$ . The sequence of equations above also proves the other direction; indeed, if  $\pi \neq \mathbf{k}^\top \mathbf{u} \otimes \mathbf{u}$ , then  $\hat{\pi} \neq \mathbf{k}^\top (\mathbf{u} + \mathbf{D}\hat{\mathbf{r}}) \otimes (\mathbf{u} + \mathbf{D}\hat{\mathbf{r}})$ .

**Soundness.** We recall that  $\mathbf{D} \in \mathbb{Z}_q^{n \times d}$ ,  $\bar{\mathbf{D}}_i \in \mathbb{Z}_q^{n \times d'}$ . If we only consider the view of the adversary given the verification key and the outputs of the simulation oracle we have that the proving key is uniformly distributed over a set of cardinality  $q^{n^2 - 2nd - 2d'^2}$ . Therefore, we require that  $n^2 > 2n \cdot d + 2d'^2$  holds.

To see this, think of  $\mathbf{k}$  as a formal variable and notice that publishing  $\mathbf{k}^\top (\mathbf{D} \otimes \mathbf{I}_n)$  counts for  $n \cdot d$  equations; also,  $\mathbf{k}^\top (\mathbf{I}_n \otimes \mathbf{D})$  counts for  $n \cdot d$  equations which in total gives us  $2n \cdot d$  equations. Moreover, in order to simulate proofs for  $[\mathbf{u}]_1 \in \text{span}([\mathbf{D}_i])$  the oracle gives away, at the worst case, the equations  $\mathbf{k}^\top (\bar{\mathbf{D}}_i \otimes \bar{\mathbf{D}}_i)$  which count for  $d'^2$  equations for each  $i \in \{0, 1\}$  which sum up to  $2d'^2$  equations in total. Indeed, expanding  $\mathbf{k}^\top (\mathbf{D}_i \otimes \mathbf{D}_i)$ , we obtain  $\mathbf{k}^\top (\mathbf{D} \otimes \mathbf{D} \parallel \bar{\mathbf{D}}_i \otimes \mathbf{D} \parallel \mathbf{D} \otimes \bar{\mathbf{D}}_i \parallel \bar{\mathbf{D}}_i \otimes \bar{\mathbf{D}}_i)$ . Now  $\mathbf{k}^\top (\bar{\mathbf{D}}_i \otimes \mathbf{D})$  and  $\mathbf{k}^\top (\mathbf{D} \otimes \bar{\mathbf{D}}_i)$  can be computed given the proving key and  $\mathbf{D}_0, \mathbf{D}_1$ . In fact, when we compute

$\mathbf{k}^\top (\mathbf{D} \otimes \mathbf{I}) (\mathbf{I} \otimes \bar{\mathbf{D}}_i)$ , we obtain  $\mathbf{k}^\top (\mathbf{D}\mathbf{I} \otimes \mathbf{I}\bar{\mathbf{D}}_i) = \mathbf{k}^\top (\mathbf{D} \otimes \bar{\mathbf{D}}_i)$ . And in a similar way, we can compute  $\mathbf{k}^\top (\bar{\mathbf{D}}_i \otimes \mathbf{D})$ . In total, we are giving up  $2n \cdot d + 2d'^2$  equations and the length of our key  $k$  is  $n^2$ .

Notice that the adversary can gather additional information about the proving key  $\mathbf{k}$  through the verification oracle. Indeed, whenever it sends a query  $([\mathbf{u}]_1, \pi)$  with  $[\mathbf{u}]_1 \in \mathcal{L}^{\text{ver}} \setminus \mathcal{L}^{\text{snd}}$  either it wins the security game or the adversary learns that  $\pi \neq \mathbf{k}^\top [\mathbf{u}]_1 \otimes [\mathbf{u}]_1$ .

Consider the hybrid experiment  $\mathbf{H}_j$  where the first  $j$ -th queries  $([\mathbf{u}]_1, \pi)$  to the verification oracle with  $[\mathbf{u}]_1 \notin \mathcal{L}^{\text{snd}}$  are answered with 0, in particular, the bit  $b$  is left unmodified, while the remaining queries are handled as in the soundness experiment. Clearly,  $\mathbf{H}_0$  is the original experiment, while  $\mathbf{H}_Q$  where  $Q$  is an upper bound on the number of verification oracle queries made by the adversary is a trivial experiment where the adversary cannot win (since the bit  $b$  will never be set to 1), thus  $\Pr[\mathbf{H}_Q = 1] = 0$ . The distinguishing event between two consecutive hybrids is the event that the adversary wins the soundness experiment at the  $j$ -th query, which happens with probability  $1/q^{n^2 - 2nd + 2d'^2} \leq 1/q$ , as it is the same as the event of guessing a uniformly random vector from a subspace of dimension  $n^2 - 2nd + 2d'^2$  of  $\mathbb{Z}_q^{n^2}$ , thus  $\Pr[\mathbf{H}_j = 1] \leq \Pr[\mathbf{H}_{j+1} = 1] + 1/q$ . Finally, by the triangular equation and noticing that  $Q$  is polynomial in the security parameter we can conclude our proof of soundness.

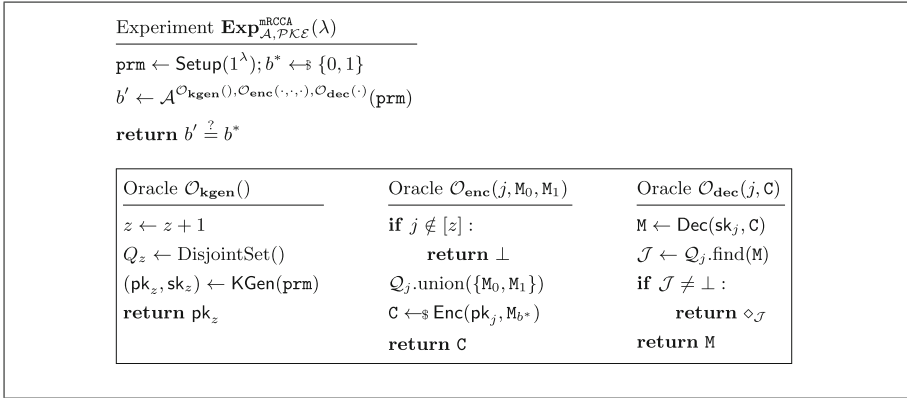
**Derivation Privacy and Zero-Knowledge.** The scheme is perfectly derivation private and zero-knowledge. For the former, notice that, for any  $\hat{\mathbf{r}}$ , we have that  $\text{PPrv}(ppk, [\mathbf{u} + \mathbf{D}\hat{\mathbf{r}}]_1, \mathbf{r} + \hat{\mathbf{r}}) = \mathbf{k}^\top [\mathbf{D} \otimes \mathbf{D}]_T \cdot ((\mathbf{r} + \hat{\mathbf{r}}) \otimes (\mathbf{r} + \hat{\mathbf{r}})) = \text{PEvl}(ppk, \pi, \hat{\mathbf{r}})$ . For the latter, given an instance  $[\mathbf{u}]_1$  such that  $[\mathbf{u}]_1 = [\mathbf{D}]_1 \mathbf{r}$ , we have that  $\text{PSim}(psk, [\mathbf{u}]_1) = \mathbf{k}^\top ([\mathbf{u}]_1 \otimes [\mathbf{u}]_1) = \mathbf{k}^\top ([\mathbf{D}\mathbf{r}]_1 \otimes [\mathbf{D}\mathbf{r}]_1) = \text{PPrv}(ppk, [\mathbf{u}]_1, \mathbf{r})$ .

### 4 Rand RCCA PKE for Multi-users and Multi-Ciphertexts

A re-randomizable PKE (Rand-PKE) scheme  $\mathcal{PK}\mathcal{E}$  is a tuple of five algorithms: (i) The algorithm **Setup** upon input the security parameter  $1^\lambda$  produces public parameters  $\text{prm}$  which include the description of the message and ciphertext space  $\mathcal{M}, \mathcal{C}$ ; (ii) The algorithm **KGen** upon input  $\text{prm}$ , outputs a key pair  $(\text{pk}, \text{sk})$ ; (iii) The algorithm **Enc** upon inputs  $\text{pk}$  and a message  $\text{M} \in \mathcal{M}$ , outputs a ciphertext  $\text{C} \in \mathcal{C}$ ; (iv) The algorithm **Dec** upon input  $\text{sk}$  and a ciphertext  $\text{C}$ , outputs a message  $\text{M} \in \mathcal{M}$  or an error symbol  $\perp$ ; (v) The algorithm **Rand** upon inputs  $\text{pk}$  and a ciphertext  $\text{C}$ , outputs another ciphertext  $\text{C}'$ .

**Definition 8 (multi-user and multi-ciphertext Replayable CCA Security).** Consider the experiment  $\text{Exp}^{\text{mRCCA}}$  in Fig. 2, with parameters  $\lambda$ , an adversary  $\mathcal{A}$ , and a PKE scheme  $\mathcal{PK}\mathcal{E}$ . We say that  $\mathcal{PK}\mathcal{E}$  is indistinguishable secure under replayable chosen-ciphertext attacks in the multi-user and multi-ciphertext setting (*mRCCA-secure*) if for any PPT adversary  $\mathcal{A}$ :

$$\text{Adv}_{\mathcal{A}, \mathcal{PK}\mathcal{E}}^{\text{mRCCA}}(\lambda) := \left| \Pr[\text{Exp}_{\mathcal{A}, \mathcal{PK}\mathcal{E}}^{\text{mRCCA}}(\lambda) = 1] - \frac{1}{2} \right| \in \text{negl}(\lambda).$$



**Fig. 2.** The multi-user and multi-ciphertext RCCA Security Experiment.

In Fig. 2, for each user  $j$  we define  $\mathcal{Q}_j$  to be a partition of the set of the challenge messages sent to the encryption oracle for the user  $j$ . To do so we use the classical “Disjoint-Set” (also called “Union-Find”) data structure from Tarjan [34]. Whenever two challenge messages are submitted to the encryption oracle, indeed, we merge the sets to which they belong so that a future call to the guarded decryption oracle behaves consistently. This allows us to express in Fig. 2 the syntax of the encryption and the guarded decryption oracle in terms of three operations: DisjointSet() that allows initializing the partition (initially empty), union( $S$ ) that adds to the partition the minimal subset of the challenge messages that contains the messages in  $S$  meanwhile maintaining invariant the partition property (i.e. a collection of disjoint sets), and find( $M$ ) that returns the set in the partition where  $M$  belongs to, or  $\perp$  if  $M$  is not in the set of challenge messages of the user  $j$ . We confirm that our definition is indeed the right multi-user and multi-ciphertext extension of the IND-RCCA definition of [7] by showing that our definition tightly implies the UC-RCCA definition of the same paper<sup>3</sup>. For space reasons, we recall the definition of the ideal functionality  $\mathcal{F}_{\text{RPKE}}$  which formalizes the notion of replay security for PKE scheme in the universal composability model in [14], where we also give the proof of the theorem below.

**Theorem 2.** *Let  $\mathcal{PKE}$  be a PKE scheme with message space  $\mathcal{D}$ . There exists a simulator  $S$  such that for any static-corruption environment  $\mathcal{Z}$  with running time  $T_{\mathcal{Z}}$  there exists an adversary  $\mathcal{B}$  whose running time is  $O(T_{\mathcal{Z}}(\lambda))$  such that:*

$$\left| \Pr[\text{REAL}_{\mathcal{Z}, \Pi_{\mathcal{PKE}}}(\lambda) = 1] - \Pr[\text{IDEAL}_{\mathcal{Z}, S}^{\mathcal{F}_{\text{RPKE}}}(\lambda) = 1] \right| \leq 2\text{Adv}_{\mathcal{B}, \mathcal{PKE}}^{\text{mucc-RCCA}}(\lambda) + \frac{T_{\mathcal{Z}}}{|\mathcal{D}|}$$

For space reasons, we only informally introduce the notions of perfect re-randomizability and public verifiability, and give more details in [14]. For the notion of perfect re-randomizability, we consider the definition given in [13] which consists of three conditions: (i) the re-randomization of a valid ciphertext and a

<sup>3</sup> In [7], the IND-RCCA notion implies the UC-RCCA notion with a loss of security that is proportional to the running time of the environment.

fresh ciphertext (for the same message) are equivalently distributed; (ii) the re-randomization procedure maintains correctness, i.e., the randomized ciphertext and the original decrypt to the same value, and in particular, invalid ciphertexts keep being invalid; (iii) it is hard to find a valid ciphertext that is not in the support of the encryption scheme. A PKE scheme is publicly verifiable if the validity of the ciphertexts can be checked only using public material.

## 5 Our Rand-RCCA PKE Scheme

<div style="border-bottom: 1px solid black; margin-bottom: 10px;"> <b>Setup</b>(<math>1^\lambda</math>)                 </div> $\text{prm}_G = (q, \mathbb{G}_1, \mathbb{G}_T, e, \mathcal{P}_1) \leftarrow \text{GGen}(1^\lambda)$ $\mathcal{M} \leftarrow \mathbb{G}_1; \mathcal{C} \leftarrow \mathbb{G}_1^{n+2} \times \mathbb{G}_T \times \mathcal{P}$ $\text{prm} \leftarrow (\text{prm}_G, \mathcal{M}, \mathcal{C})$ <b>return</b> prm <div style="border-top: 1px solid black; margin-top: 10px;"> <b>KGen</b>(prm)                 </div> $\mathbf{D} \leftarrow \mathcal{D}_{n,d}, \mathbf{a} \leftarrow \mathbb{Z}_q^n$ $\mathbf{D}^* \leftarrow (\mathbf{D}^\top, (\mathbf{a}^\top \mathbf{D})^\top)^\top$ $\mathbf{f} \leftarrow \mathbb{Z}_q^n, \mathbf{F} \leftarrow \mathbb{Z}_q^{n \times n+1}$ $\text{pars} \leftarrow (\text{prm}_G, [\mathbf{D}]_1)$ $\text{ppk}, \text{psk} \leftarrow \text{PGen}(\text{pars})$ $\text{pk} \leftarrow ([\mathbf{D}^*]_1, [\mathbf{f}^\top \mathbf{D}]_T, [\mathbf{F}^\top \mathbf{D}]_1, [\mathbf{F} \mathbf{D}^*]_1, \text{ppk})$ $\text{sk} \leftarrow (\mathbf{a}, \mathbf{f}, \mathbf{F}, \text{psk})$ <b>return</b> (pk, sk)	<div style="border-bottom: 1px solid black; margin-bottom: 10px;"> <b>Enc</b>(pk, <math>[M]_1</math>)                 </div> $\mathbf{r} \leftarrow \mathbb{Z}_q^d$ $[\mathbf{u}]_1 \leftarrow [\mathbf{D}]_1 \cdot \mathbf{r}, \pi \leftarrow \text{PPrv}(\text{ppk}, [\mathbf{u}]_1, \mathbf{r})$ $[p]_1 \leftarrow [\mathbf{a}^\top \mathbf{D}]_1 \cdot \mathbf{r} + [M]_1$ $[\mathbf{x}]_1 \leftarrow ((\mathbf{u}^\top)_1, [p]_1)^\top$ $[y]_T \leftarrow \left( [\mathbf{f}^\top \mathbf{D}]_T + e([\mathbf{x}]_1^\top, [\mathbf{F}^\top \mathbf{D}]_1) \right) \cdot \mathbf{r}$ <b>return</b> C := $([\mathbf{x}]_1, [y]_T, \pi)$ <div style="border-top: 1px solid black; margin-top: 10px;"> <b>Dec</b>(sk, C)                 </div> <b>parse</b> C as $([\mathbf{x}]_1, [y]_T, \pi)$ <b>parse</b> $[\mathbf{x}^\top]_1$ as $([\mathbf{u}^\top]_1, [p]_1)$ $[M]_1 \leftarrow [p]_1 - [\mathbf{a}^\top \mathbf{u}]_1$ $[y']_T \leftarrow \mathbf{f}^\top e([\mathbf{1}]_1, [\mathbf{u}]_1) + e(\mathbf{F}[\mathbf{x}]_1, [\mathbf{u}]_1)$ $b_1 \leftarrow [y']_T \stackrel{?}{=} [y]_T, b_2 \leftarrow \text{PVer}(\text{psk}, [\mathbf{u}]_1, \pi)$ <b>if</b> $b_1 \wedge b_2$ <b>return</b> $[M]_1$ <b>else</b> $\perp$
<div style="border-bottom: 1px solid black; margin-bottom: 10px;"> <b>Rand</b>(pk, C)                 </div> <b>parse</b> C as $([\mathbf{x}]_1, [y]_T, \pi)$ , <b>parse</b> $[\mathbf{x}^\top]_1$ as $([\mathbf{u}^\top]_1, [p]_1)$ $\hat{\mathbf{r}} \leftarrow \mathbb{Z}_q^d, [\hat{\mathbf{x}}]_1 \leftarrow [\mathbf{x}]_1 + [\mathbf{D}^*]_1 \cdot \hat{\mathbf{r}}$ $[\hat{y}]_T \leftarrow [y]_T + [\mathbf{f}^\top \mathbf{D}]_T \cdot \hat{\mathbf{r}} + e([\mathbf{x}]_1, [\mathbf{F}^\top \mathbf{D}]_1 \cdot \hat{\mathbf{r}}) + e([\mathbf{F} \mathbf{D}^*]_1 \cdot \hat{\mathbf{r}}, [\hat{\mathbf{u}}]_1)$ $\hat{\pi} \leftarrow \text{PEvl}(\text{ppk}, [\mathbf{u}]_1, \pi, \hat{\mathbf{r}})$ <b>return</b> $\hat{C} := ([\hat{\mathbf{x}}]_1, [\hat{y}]_T, \hat{\pi})$	

**Fig. 3.** Rand-RCCA PKE scheme  $\mathcal{PKE}$  based on the  $\mathcal{D}_{n,d}$ -MDDH assumption in type-1 bilinear groups.  $\mathcal{P}$  is the support of the proofs for **PS**.

We present our scheme in Fig. 3. With the goal of improving readability for developers, all the operations (and in particular the pairing operations) in the figure are described explicitly using  $e$  for the pairing and  $\cdot$  for the exponentiations. The scheme can be summarized as a type-1 pairing group version of the scheme in [13] where we additionally append a benign proof to prove almost tight-security. The main technical component from [13] to obtain RCCA security is the *consistency check* at decryption time which checks that  $[y]_T \stackrel{?}{=} \mathbf{f}^\top [\mathbf{u}]_T + [\mathbf{x}]_1^\top \mathbf{F}^\top [\mathbf{u}]_1$

**Perfect Re-Randomizability.** The proof of perfect re-randomizability follows from [13] and the derivation privacy of **PS**. Here we highlight the following lemma, whose proof is in [14].

**Lemma 2.** *For any  $[\mathbf{x}]_1$  and  $\hat{\mathbf{r}}$ , let  $[\hat{\mathbf{x}}]_1 = [\mathbf{x}]_1 + [\mathbf{D}^*]_1 \hat{\mathbf{r}}$ , we have that:*

$$\begin{aligned}
 (\mathbf{f}^\top + [\hat{\mathbf{x}}]_1^\top \mathbf{F}^\top)[\hat{\mathbf{u}}]_1 &= (\mathbf{f}^\top + [\mathbf{x}]_1^\top \mathbf{F}^\top)[\mathbf{u}]_1 + [\mathbf{f}^\top \mathbf{D}]_T \cdot \hat{\mathbf{r}} \\
 &\quad + e([\mathbf{x}]_1, [\mathbf{F}^\top \mathbf{D}]_1 \cdot \hat{\mathbf{r}}) + e([\mathbf{F} \mathbf{D}^*]_1 \cdot \hat{\mathbf{r}}, [\hat{\mathbf{u}}]_1)
 \end{aligned}$$

The correctness of  $\mathcal{PK}\mathcal{E}$  follows from the lemma above and the fact that **PS** and **PEvI** form a malleable proof system. More details are in [14].

**Security.** We prove that the security of the scheme reduces to the  $\mathcal{D}_{n,d}$ -MDDH assumption. Below we state the main theorem.

**Theorem 3.** *For every PPT adversary  $\mathcal{A}$  that makes at most  $Q_{\text{Enc}}$  encryption and  $Q_{\text{Dec}}$  decryption queries, there exist adversaries  $\mathcal{B}^{\text{mddh}}$ ,  $\mathcal{B}^{\text{snd}}$  with similar running time  $T(\mathcal{B}^{\text{mddh}}) \approx T(\mathcal{B}^{\text{snd}}) \approx T(\mathcal{A}) + (Q_{\text{Enc}} + Q_{\text{Dec}}) \cdot \text{poly}(\lambda)$ , where  $\text{poly}(\lambda)$  is a polynomial independent of  $T(\mathcal{A})$ , and such that*

$$\begin{aligned}
 \text{Adv}_{\mathcal{A}, \mathcal{PK}\mathcal{E}}^{\text{RCCA}}(\lambda) &\leq O(d \log Q_{\text{Enc}}) \cdot \text{Adv}_{\mathbf{G}_1, \mathcal{D}_{n,d}, \mathcal{B}^{\text{mddh}}}^{\text{MDDH}}(\lambda) \\
 &\quad + \log Q_{\text{Enc}} \cdot \text{Adv}_{\mathcal{B}^{\text{snd}}, \mathbf{PS}}^{\text{snd}}(\lambda) + O\left(\frac{n^2 Q_{\text{Dec}} Q_{\text{Enc}} \log Q_{\text{Enc}}}{q}\right).
 \end{aligned}$$

*Proof.* We give a proof only for the single-user, multi-ciphertext case, i.e. when the adversary calls the key generation oracle only once. The proof can be easily generalized<sup>4</sup> to the multi-user case almost equivalently to [4, 18].

To simplify the notation, since we are in the single-user setting, we omit the index  $j$  (which specifies the user) from both encryption and decryption queries. We let  $\mathbf{G}_0$  be the  $\text{Exp}_{\mathcal{A}, \mathcal{PK}\mathcal{E}}^{\text{mRCCA}}$  experiment, and we denote with  $\epsilon_i$  the advantage of  $\mathcal{A}$  to win  $\mathbf{G}_i$ , i.e.  $\epsilon_i := |\Pr[\mathbf{G}_i = 1] - \frac{1}{2}|$ .

The games keep track of the number of challenge ciphertexts produced. Specifically, let  $\text{ctr}$  be a variable that counts the number of challenge ciphertexts output by the encryption oracle:  $\text{ctr}$  is set to 0 at the beginning of the games and, whenever the adversary calls the encryption oracle, it is increased.

**Game  $\mathbf{G}_1$ .** This game is identical to the previous one, but the encryption oracle computes the values  $[y]_T$  and  $[p]$  using secret keys (instead of public keys). Specifically, upon the  $j$ -th query to the encryption oracle, the game computes the ciphertext  $\mathbf{C}_j = ([\mathbf{x}_j], [y_j]_T, \pi_j)$  as described by the encryption procedure,

<sup>4</sup> We rely on the self-reducibility of the MDDH assumption: in particular, we can generate  $m$  different matrices  $\mathbf{D}_j$  (one for each user) from one single challenge of the (many-fold) MDDH-assumption and adapt accordingly the ciphertexts, namely, by mapping the ciphertext for the  $j$ -th user through the same linear transformation that maps the MDDH-challenge matrix to the matrix  $\mathbf{D}_j$ .



but where we compute  $[y_j]_T \leftarrow \mathbf{f}^\top[\mathbf{u}_j] + [\mathbf{x}_j]^\top \mathbf{F}^\top[\mathbf{u}_j]$  and  $[p_j] \leftarrow \mathbf{a}^\top[\mathbf{u}_j] + [\mathbf{M}_{j,b^*}]$ . By linearity, this game is perfectly equivalent to the previous one, thus  $\epsilon_1 = \epsilon_0$ .

**Game  $\mathbf{G}_2$ .** This game is identical to the previous one, but the encryption oracle simulates the benign proofs  $\pi$ . We rely on the perfect zero-knowledge of the benign proof system. The reduction is standard, therefore we omit it. Since the proof system satisfies perfect zero-knowledge we have  $\epsilon_2 = \epsilon_1$ .

**Game  $\mathbf{G}_3$ .** At the very beginning, the game additionally samples matrices  $\bar{\mathbf{D}}_b \leftarrow \mathbb{Z}_q^{n \times d}$  for  $b \in \{0, 1\}$ , and sets  $\mathbf{D}_b \leftarrow (\mathbf{D} | \bar{\mathbf{D}}_b)$ . The encryption oracle in this game samples  $[\mathbf{u}]$  from the span of  $[\mathbf{D}_0]$ . We apply a standard reduction to the  $Q_{\text{Enc}}$ -fold  $\mathcal{D}_{n,d}$ -MDDH assumption, twice, and we prove that no adversary can distinguish this game from the previous one: we first tightly switch the vectors in the challenge ciphertexts from the span of  $[\mathbf{D}]$  to uniformly random vectors of  $\mathbb{G}_1^n$ ; next, we use the  $Q_{\text{Enc}}$ -fold  $\mathcal{D}_{n,2d}$ -MDDH assumption to switch these vectors from random to the span of  $[\mathbf{D}_0]$ . The proof of this step is standard: in [14], we show how we can build adversaries  $\mathcal{B}, \mathcal{B}'$  such that

$$|\epsilon_3 - \epsilon_2| \leq \mathbf{Adv}_{\mathbb{G}_1, \mathcal{D}_{n,d}, \mathcal{B}}^{Q_{\text{Enc}}\text{-MDDH}}(\lambda) + \mathbf{Adv}_{\mathbb{G}_1, \mathcal{D}_{n,2d}, \mathcal{B}'}^{Q_{\text{Enc}}\text{-MDDH}}(\lambda)$$

**Game  $\mathbf{G}_4$ .** In this experiment, we add an explicit check to the decryption oracle. First recall that  $\mathbf{D}^*$  is defined in Fig. 3 as the matrix whose first  $n$  rows are equal to  $\mathbf{D}$  and last row is equal to  $\mathbf{a}^\top \mathbf{D}$ . Upon query  $\mathbf{C} := ([\mathbf{x}], [y]_T, \pi)$  to the decryption oracle, where  $[\mathbf{x}]^\top := ([\mathbf{u}]^\top, [p])$ , the oracle additionally checks that:

$$\mathbf{u} \in \text{span}(\mathbf{D}) \vee \exists j : \mathbf{D}^{*\perp} \mathbf{x}_j = \mathbf{D}^{*\perp} \mathbf{x} \quad (3)$$

where  $\mathbf{D}^{*\perp} \mathbf{D}^* = 0$ , and  $\mathcal{Q}_{\text{Enc}} = \{\mathbf{C}_j = ([\mathbf{x}_j], [y_j]_T, \pi_j) : j \leq [\text{ctr}]\}$  is the set of challenge ciphertexts. If the condition holds, the decryption oracle proceeds by running the decryption procedure as usual, otherwise it returns  $\perp$  to the adversary. We notice that the new condition can be checked efficiently since we know  $\mathbf{D} \in \mathbb{Z}_q^{n \times d}$  and  $\mathbf{a} \in \mathbb{Z}_q^n$ .

The distinguishing event between  $\mathbf{G}_4$  and  $\mathbf{G}_3$  is that the adversary queries the decryption oracle with a ciphertext that would not decrypt to  $\perp$  (according to the original decryption rules of  $\mathbf{G}_3$ ), but where Eq. (3) holds. We call such query to the decryption oracle a “critical query”, i.e. a decryption query where:

- $[\mathbf{u}] \notin \text{span}([\mathbf{D}])$  and  $\forall j : \mathbf{D}^{*\perp} \mathbf{x}_j \neq \mathbf{D}^{*\perp} \mathbf{x}$  (the latter implies that  $[\mathbf{u}]$  is not the result of an honest rerandomization of a previous challenge ciphertext)
- $\pi$  is valid, and  $[y]_T = \mathbf{f}^\top[\mathbf{u}]_T + [\mathbf{x}]^\top \mathbf{F}^\top[\mathbf{u}]$ , i.e., the consistency check holds.

For this step, we refer to Lemma 3.

**Game  $\mathbf{G}_5$ .** This game is equivalent to  $\mathbf{G}_4$ , but we modify the rules of the decryption oracle once again. For any  $j$ , let  $\mathbf{M}_{j,0}$  and  $\mathbf{M}_{j,1}$  be the challenge messages queried by  $\mathcal{A}$  at the  $j$ -th query to the encryption oracle. Upon decryption query  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$ , if  $\exists j : \mathbf{D}^{*\perp} \mathbf{x}_j = \mathbf{D}^{*\perp} \mathbf{x}$  where recall  $\mathcal{Q}_{\text{Enc}} = \{([\mathbf{x}_j], [y_j], \pi_j) : j \leq \text{ctr}\}$ , and both the proof  $\pi$  verifies and the consistency check holds, then the decryption oracle immediately returns the symbol  $\diamond_{\mathcal{J}}$  where  $\mathcal{J} \leftarrow \mathcal{Q}.\text{find}(\mathbf{M}_{j,0})$ .

Notice that we can rewrite the decryption procedure as  $\mathbf{M} = (-\mathbf{a}^\top, 1)[\mathbf{x}]$ . We observe that the vector  $(-\mathbf{a}^\top, 1)$  is in the span of  $\mathbf{D}^{*\perp}$ , since it holds that  $(-\mathbf{a}^\top, 1)\mathbf{D}^* = -\mathbf{a}^\top\mathbf{D} + \mathbf{a}^\top\mathbf{D} = 0$ . Thus, at any decryption query, if  $\mathbf{D}^{*\perp}\mathbf{x}_j = \mathbf{D}^{*\perp}\mathbf{x}_j$  for some challenge ciphertext  $\mathbf{C}_j$  then  $(-\mathbf{a}^\top, 1)[\mathbf{x}_j] = (-\mathbf{a}^\top, 1)[\mathbf{x}]$ , and therefore the decryption oracle would compute the message  $M_{j,b^*}$  and output the symbol  $\diamond_{\mathcal{J}}$ , where  $\mathcal{J} = \mathcal{Q}.\text{find}(M_{j,b^*})$ . Moreover, notice that  $\mathcal{Q}.\text{find}(M_{j,b^*}) = \mathcal{Q}.\text{find}(M_{j,0})$  by definition of the security experiment. This shows that  $\epsilon_5 = \epsilon_4$ .

**Game  $\mathbf{G}_6$ .** In this last step, we encrypt random messages. Formally, at the  $j$ -th encryption query the oracle (on input messages  $M_{j,0}, M_{j,1}$ ) encrypts the message  $M_{j,b^*} + \mathbf{R}_j$ , where  $\mathbf{R}_j$  is random. Clearly, it holds that  $\epsilon_6 = 0$  as in fact, because of the change introduced in  $\mathbf{G}_6$ , the ciphertexts are independent of the challenge bit  $b^*$ , and, by the changes introduced in  $\mathbf{G}_4$  and  $\mathbf{G}_5$ , the decryption queries are independent of the challenge bit. We prove that  $\mathbf{G}_5$  and  $\mathbf{G}_6$  are indistinguishable, as this step is almost the same as in [18], we defer its proof to [14].

**Lemma 3.** *For any PPT adversary  $\mathcal{A}$ , we build PPT adversaries  $\mathcal{B}, \mathcal{B}'$  with running times similar to  $\mathcal{A}$  such that:*

$$|\epsilon_3 - \epsilon_4| \leq O(d \log Q_{\text{Enc}}) \text{Adv}_{\mathbb{G}_1, \mathcal{D}_{n,d}, \mathcal{B}}^{\text{MDDH}}(\lambda) + \log Q_{\text{Enc}} \text{Adv}_{\mathcal{B}'}^{\text{snd}}(\lambda) + O\left(\frac{n^2 Q_{\text{Dec}} Q_{\text{Enc}} \log Q_{\text{Enc}}}{q}\right)$$

*Proof.* We denote the probability that the adversary  $\mathcal{A}$  wins game  $\mathbf{H}_x$  by  $\epsilon_{\mathbf{H}_x}$ . In the following, we will bound  $\epsilon_{\mathbf{H}_0}$  via a sequence of games.

**Hybrid  $\mathbf{H}_0$ .** This hybrid is the same as  $\mathbf{G}_3$  but immediately outputs 1 if the adversary makes a “critical query”. Specifically, the hybrid executes  $\mathbf{G}_3$  but the decryption oracle upon input  $\mathbf{C}$  parses it as  $([\mathbf{x}], [y]_T, \pi)$  and checks that Eq. (3) holds; if it holds, the decryption oracle continues as before. Otherwise, returns the message “critical”, and  $\mathbf{H}_0$  stops the interaction, immediately returning 1. Since the hybrid outputs 1 when the distinguishing event between  $\mathbf{G}_3$  and  $\mathbf{G}_4$  happens, we have that  $|\epsilon_3 - \epsilon_4| \leq \epsilon_{\mathbf{H}_0}$ . Also notice that the checks in Eq. (3) can be efficiently performed given the knowledge of  $\mathbf{D}$ .

**Hybrid  $\mathbf{H}_1$ .** This hybrid is preparatory for the next one. We inject randomness into the encryption/decryption keys, adding a vector  $(\mathbf{zD}^\perp)$  to the secret key  $\mathbf{f}^\top$ , common to all the encryption queries, where  $\mathbf{z} \in \mathbb{Z}_q^{n-d}$ . Specifically, at the very beginning of the experiment we sample the vector  $\mathbf{z} \leftarrow \mathbb{Z}_q^{n-d}$ , we sample  $\mathbf{f}$  and compute the public key material  $[\mathbf{f}^\top \mathbf{D}]$  and moreover:

- The encryption oracle, at the  $j$ -th query, computes the values  $[y_j]_T$  as follows:

$$[y_j]_T \leftarrow (\mathbf{f}^\top + \mathbf{zD}^\perp)[\mathbf{u}_j]_T + [\mathbf{x}_j]^\top \mathbf{F}^\top [\mathbf{u}_j]$$

- Similarly, the decryption oracle, upon input the ciphertext  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$  computes the bit  $b_1$  (i.e. the bit of the consistency check) by computing the value  $[y']_T$  and checking if  $[y]_T \stackrel{?}{=} [y']_T$  where  $[y']_T$  is computed as:

$$[y']_T \leftarrow (\mathbf{f}^\top + \mathbf{zD}^\perp)[\mathbf{u}]_T + [\mathbf{x}]^\top \mathbf{F}^\top [\mathbf{u}]$$

These new rules do not change the view of the adversary since both  $\mathbf{f}^\top$  and  $\mathbf{f}^\top + \mathbf{zD}^\perp$  are uniformly distributed over  $\mathbb{Z}_q^{1 \times n}$  given the public key material  $[\mathbf{f}^\top \mathbf{D}]$ . Thus we obtain  $\epsilon_{\mathbf{H}_1} = \epsilon_{\mathbf{H}_0}$ .

**Hybrid  $\mathbf{H}_2$ .** Let  $P : \{0, 1\}^* \rightarrow \mathbb{Z}_q^{1 \times n-d}$  be a uniformly random function. In this hybrid we use the following rules for encryption and decryption:

- The encryption oracle, at the  $j$ -th query, computes the values  $[y_j]_T$  as follows:

$$[y_j]_T \leftarrow (\mathbf{f}^\top + P(j) \mathbf{D}^\perp)[\mathbf{u}_j]_T + [\mathbf{x}_j]^\top \mathbf{F}^\top [\mathbf{u}_j]$$

- For each decryption oracle query, we first define a set  $\mathcal{S}$  over which the decryption oracle iterates to test the consistency check. The definition of the set  $\mathcal{S}$  is carefully crafted to define the behavior of the hybrid experiment in case of *replay attack* from the adversary

Recall that `ctr` counts the number of challenge ciphertexts output by the encryption oracle and that  $\mathcal{Q}_{\text{Enc}} = \{\mathbf{C}_j = ([\mathbf{x}_j], [y_j]_T, \pi_j) : j \leq \text{ctr}\}$ . Upon input the ciphertext  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$ , the decryption oracle first sets:

$$\begin{aligned} \mathcal{S} &:= \{j\} && \text{if } \exists j \leq \text{ctr} : \mathbf{D}^{*\perp}[\mathbf{x}] = \mathbf{D}^{*\perp}[\mathbf{x}_j] \\ \mathcal{S} &:= \{j : j \leq \text{ctr}\} && \text{otherwise} \end{aligned}$$

then it computes the bit  $b_1$  (i.e. the bit of the consistency check for  $\mathbf{C}$ , see Fig. 3) differently by checking that

$$\exists j \in \mathcal{S} : [y]_T \stackrel{?}{=} (\mathbf{f}^\top + P(j) \mathbf{D}^\perp)[\mathbf{u}]_T + [\mathbf{x}]^\top \mathbf{F}^\top [\mathbf{u}].$$

Moving from  $\mathbf{H}_1$  to  $\mathbf{H}_2$  requires a series of hybrids  $\mathbf{H}_{1,i,i'}$ ,  $i \in [\log(Q_{\text{Enc}})]$ ,  $i' \in [6]$ . We give in [14] the formal definitions of all these hybrids, and we highlight their differences.

**Hybrid  $\mathbf{H}_{1,i,0}$ .** Let  $P_i$  be a random function that takes in input strings of length  $i$  (for  $i = 0$ , we can imagine this as a constant function defined on the empty string) and returns row vectors of length  $n - d$ .

- On input the  $j$ -th query, the encryption oracle samples  $[\mathbf{u}_j]$  from the span of  $[\mathbf{D}_0]$ . The element  $[y_j]_T$  is computed as

$$[y_j]_T \leftarrow (\mathbf{f} + P_i(j_i) \mathbf{D}^\perp)[\mathbf{u}_j] + [\mathbf{x}_j]^\top \mathbf{F}^\top [\mathbf{u}_j].$$

- Upon input the ciphertext  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$ , define:

$$\begin{aligned} \mathcal{S} &:= \{j_i\} && \text{if } \exists j \leq \text{ctr} : \mathbf{D}^{*\perp}[\mathbf{x}] = \mathbf{D}^{*\perp}[\mathbf{x}_j] \\ \mathcal{S} &:= \{j_i : j \leq \text{ctr}\} && \text{otherwise} \end{aligned}$$

it then executes the same code of the previous hybrid.

When  $i = 0$ , for any value  $j$  the string  $j|_0$  is equal to the empty string, thus, in  $\mathbf{H}_{1,0,0}$ , the random function  $P_0$  is always called on input the empty string. In particular, either when  $\mathbf{D}^{*\perp}[\mathbf{x}] = \mathbf{D}^{*\perp}[\mathbf{x}_j]$  holds or when it does not, the consistency check performed is exactly the same. Thus the difference between hybrid  $\mathbf{H}_{1,0,0}$  and  $\mathbf{H}_1$  is only syntactical.

**Hybrid  $\mathbf{H}_{1,i,1}$ .** This hybrid is equivalent to the previous one, but here the encryption oracle, on input the  $j$ -th query, generates  $[\mathbf{u}_j]$  in the span of  $[\mathbf{D}_{j[i+1]}]$ . We rely on the MDDH assumption to prove indistinguishability between the two hybrids. We proceed in two steps:

- We first switch the  $j$ -th vector  $[\mathbf{u}_j]$  computed by the encryption oracle to a vector in the span of  $[(\mathbf{D}|\mathbf{U})]$ , where  $\mathbf{U}$  is uniform over  $\mathbb{Z}_q^{n \times d}$ , if the  $(i + 1)$ -th bit of the binary representation of  $j$  is equal to 1. We call this intermediate hybrid  $\mathbf{H}_{A_i}$ .
- Finally, we switch the  $j$ -th vector  $[\mathbf{u}_j]$  computed by the encryption oracle to a vector in the span of  $[(\mathbf{D}|\bar{\mathbf{D}}_1)] = [\mathbf{D}_1]$ , if the  $(i + 1)$ -th bit of the binary representation of  $j$  is equal to 1.

First we show indistinguishability between  $\mathbf{H}_{1,i,0}$  and  $\mathbf{H}_{A_i}$ . Let  $\mathcal{B}_A$  be an MDDH-adversary receiving the  $Q_{\text{Enc}}$ -fold  $\mathcal{D}_{n,d}$ -MDDH challenge  $([\bar{\mathbf{D}}_0], [\mathbf{h}_1], \dots, [\mathbf{h}_{Q_{\text{Enc}}}]$ ) as input.  $\mathcal{B}_A$  can sample a random matrix  $\mathbf{D} \leftarrow \mathcal{D}_{n,d}$ , a random matrix  $\bar{\mathbf{D}}_1 \in \mathbb{Z}_q^{n \times d}$ , the secret material  $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ ,  $\mathbf{f} \leftarrow \mathbb{Z}_q^d$ ,  $\mathbf{F} \leftarrow \mathbb{Z}_q^{n \times n+1}$  and the secret material for the benign proof system (since  $\mathcal{B}_A$  knows  $\mathbf{D}$ , this can be easily achieved running  $\text{PGen}([\mathbf{D}])$ ). Finally,  $\mathcal{B}_A$  samples a challenge bit  $b$  and gives the public key of the scheme to  $\mathcal{A}$ .  $\mathcal{B}_A$  simulates the encryption oracle as follows. On input the  $j$ -th pair of messages  $(\mathbf{M}_0, \mathbf{M}_1)$ :

- if the  $(i+1)$ -th bit of the binary representation of  $j$  is equal to 0, the adversary sets  $[\mathbf{u}_j] \leftarrow [\mathbf{D}_0]\mathbf{r}_j$ ,
- else, samples a random vector  $\tilde{\mathbf{r}} \in \mathbb{Z}_q^d$ , and computes  $[\mathbf{u}_j] \leftarrow [\mathbf{D}]\tilde{\mathbf{r}} + [\mathbf{h}_j]$ .

Note that  $\mathcal{B}_A$  can still simulate the decryption oracle, because of the knowledge of the secret material  $\mathbf{a}, \mathbf{f}, \mathbf{F}$  and of the matrix  $\mathbf{D}$ . Since  $\mathcal{B}_A$  knows both the matrix  $\mathbf{D}$  and the vector  $\mathbf{a}$ , can always find a matrix  $\mathbf{D}^{*\perp}$  such that  $\mathbf{D}^{*\perp}\mathbf{D}^* = 0$ . This allows  $\mathcal{B}_A$  to catch critical queries. If the tuple is a real MDDH tuple, i.e.  $[\mathbf{h}_j] = [\bar{\mathbf{D}}_0]\mathbf{r}_j$ , the game described is perfectly equivalent to  $\mathbf{H}_{1,i,0}$ . Otherwise, if the challenge vectors are uniformly random, the game simulated is equivalent to  $\mathbf{H}_{A_i}$ . The next step is to switch the  $j$ -th vector  $[\mathbf{u}_j]$  computed by the encryption oracle to a vector in the span of  $[(\mathbf{D}|\bar{\mathbf{D}}_1)] = [\mathbf{D}_1]$  if the  $(i + 1)$ -th bit of the binary representation of  $j$  is equal to 1. This transformation is similar to the previous one, therefore we omit the details. Altogether, combining the previous adversaries, we obtain an adversary  $\mathcal{C}$  such that:  $|\epsilon_{\mathbf{H}_{1,i,1}} - \epsilon_{\mathbf{H}_{1,i,0}}| \leq 2(n - d)\text{Adv}_{\mathbb{G}_1, \mathcal{D}_{n,d}, \mathcal{C}}^{\text{mddh}}(\lambda) + \frac{2}{q-1}$ .

**Hybrid  $\mathbf{H}_{1,i,2}$ .** We add an explicit check to the decryption oracle. Specifically, at each decryption oracle query the hybrid additionally checks if  $\mathbf{u} \notin \text{span}(\mathbf{D}_0) \cup \text{span}(\mathbf{D}_1)$ , and if it is the case the decryption oracle returns immediately  $\perp$  to

the adversary. We rely on the soundness of the underlying benign proof system. In particular, the only condition that would allow to distinguish between this hybrid and the previous one is to query the decryption oracle with a ciphertext  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$  where:

- $\mathbf{u} \notin \text{span}(\mathbf{D}_0) \cup \text{span}(\mathbf{D}_1)$
- the decryption oracle in the hybrid  $\mathbf{H}_{1,i,1}$  would not return  $\perp$ .

For such query it holds that  $\text{PVer}(psk, [\mathbf{u}], \pi) = 1$ . We build an adversary  $\mathcal{B}$  against the  $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$ -soundness of the proof system. (Recall that  $\mathcal{L}^{\text{snd}} = \mathcal{L}^{\text{sim}} = \text{span}(\mathbf{D}_0) \cup \text{span}(\mathbf{D}_1)$ , and  $\mathcal{L}^{\text{ver}} = \mathbb{Z}_q^n$ .)

The adversary  $\mathcal{B}$  samples the secret material  $\mathbf{a}, \mathbf{f}, \mathbf{F}$ ; then, it queries the challenger to obtain the public key of the benign proof system, associated with the matrix  $\mathbf{D}$ , and finally gives  $\mathcal{A}$  all the public key material. The adversary  $\mathcal{B}$  can easily simulate the encryption oracle since it knows all the necessary information. To compute the proof  $\pi_j$  associated with the  $j$ -th encryption oracle query, it queries the simulation oracle offered by the challenger: it holds that  $\mathbf{u}_j \in \mathcal{L}^{\text{sim}}$ , for all  $j \in [Q_{\text{Enc}}]$ . When the adversary makes a decryption query,  $\mathcal{B}$  needs to verify that the proof  $\pi$  is accepted by  $\text{PVer}$ ; so, it forwards  $(\mathbf{u}, \pi)$  to the challenger. Since  $\mathcal{L}^{\text{ver}}$  is equal to  $\mathbb{Z}_q^n$ , the verification oracle always returns a verdict bit, and  $\mathcal{B}$  can proceed in the natural way the simulation of the decryption oracle. At some point  $\mathcal{B}$  queries the verification oracle with some  $([\mathbf{u}], \pi)$  such that  $\mathbf{u} \notin \text{span}(\mathbf{D}_0) \cup \text{span}(\mathbf{D}_1)$ , i.e.,  $\mathbf{u} \notin \mathcal{L}^{\text{snd}}$ , but  $\text{PVer}(psk, [\mathbf{u}], \pi) = 1$ . This is the event that lets  $\mathcal{B}$  win the soundness game. The adversary  $\mathcal{B}$  runs in time  $T(\mathcal{B}) \approx T(A) + (Q_{\text{Enc}} + Q_{\text{Dec}}) \cdot \text{poly}(\lambda)$ , where  $\text{poly}$  is a polynomial independent of  $T(A)$ . Moreover, notice that when the distinguishing event happens the adversary  $\mathcal{B}$  wins the soundness game, thus:  $|\epsilon_{\mathbf{H}_{1,i,2}} - \epsilon_{\mathbf{H}_{1,i,1}}| \leq \text{Adv}_{\mathcal{B}, \text{PS}}^{\text{snd}}(\lambda)$ .

**Hybrid  $\mathbf{H}_{1,i,3}$ .** In this hybrid, we increase the entropy of the secret keys during encryption queries.

- The encryption oracle, at the  $j$ -th query, computes the values  $[y_j]_T$  as follows:

$$[y_j]_T \leftarrow (\mathbf{f}^\top + P_{i+1}(j_{i+1}) \mathbf{D}^\perp)[\mathbf{u}_j] + [\mathbf{x}_j]^\top \mathbf{F}^\top [\mathbf{u}_j].$$

- The decryption oracle, upon input the ciphertext  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$  additionally checks that  $\exists d$  s.t.  $\mathbf{u} \in \text{span}(\mathbf{D}_d)$  and in such a case it sets:

$$\begin{aligned} \mathcal{S} &:= \{j_{|i} \parallel d\} && \text{if } \exists j \leq \text{ctr} : \mathbf{D}^{*\perp}[\mathbf{x}] = \mathbf{D}^{*\perp}[\mathbf{x}_j] \\ \mathcal{S} &:= \{j_{|i} \parallel d : j \leq \text{ctr}\} && \text{otherwise} \end{aligned}$$

and it continues executing the same code as the previous hybrid.

We prove that  $|\epsilon_{\mathbf{H}_{1,i,2}} - \epsilon_{\mathbf{H}_{1,i,3}}|$  is negligible. We first transit to an intermediate hybrid  $\mathbf{H}'_i$  where instead of using the function  $P_i(\cdot)\mathbf{D}^\perp$ , we use the function  $P'_i(\cdot) := P_i^{(0)}(\cdot)\mathbf{D}_0^\perp + P_i^{(1)}(\cdot)\mathbf{D}_1^\perp$ , where  $P_i^{(0)}$  and  $P_i^{(1)}$  are two uniformly random functions with domain  $\{0, 1\}^i$ . Notice that  $P'_i(\cdot)$  is an uniformly random function

that maps strings in  $\{0, 1\}^i$  to vectors in  $\text{rowspan}(\mathbf{D}_0^\perp) + \text{rowspan}(\mathbf{D}_1^\perp)$  while  $P_i(\cdot)\mathbf{D}^\perp$  is a uniformly random function that maps string in  $\{0, 1\}^i$  to vectors in  $\text{rowspan}(\mathbf{D}^\perp)$ . Thus the distinguishing event between  $\mathbf{H}_{i,j,2}$  and this intermediate hybrid is the event that  $\text{rowspan}(\mathbf{D}_0^\perp) + \text{rowspan}(\mathbf{D}_1^\perp) \neq \text{rowspan}(\mathbf{D}^\perp)$ . The latter event happens with probability at most  $1/q$ : in fact, the event happens if and only if the subspace  $\text{span}(\bar{\mathbf{D}}_0 \parallel \bar{\mathbf{D}}_1)$  has dimension strictly less than  $2d$  and recall that the columns of such matrices are sampled uniformly at random. Next, we define the function  $P_{i+1}^{(b)} : \{0, 1\}^{i+1} \rightarrow \mathbb{Z}_q^{1 \times (n-2d)}$ ,  $\forall b \in \{0, 1\}$ :

$$P_{i+1}^{(b)}(\mathbf{x}) = \begin{cases} P_i^{(b)}(\mathbf{x}_{|i}), & \mathbf{x}[i+1] \neq b \\ \tilde{P}_i^{(b)}(\mathbf{x}_{|i}), & \text{else} \end{cases}$$

where  $P_i, \tilde{P}_i$  are two uniformly (and independent) random functions. Notice that  $P_{i+1}^{(b)}$  is a uniformly random function.

We define a second intermediate hybrid  $\mathbf{H}'_{i+1}$  where for the encryption oracle queries instead of using the random function  $P'_i$  applied to the indexes  $j_{|i}$  we use the function  $P'_{i+1}$  applied to the indexes  $j_{|i+1}$ , and for the decryption oracle queries we use  $P'_{i+1}$  applied to  $(j_{|i} \parallel d)$ , where  $d$  is such that  $\mathbf{u}_j \in \text{span}(\mathbf{D}_d)$  (as described in the  $\mathbf{H}_{1,i,3}$ ). We show that  $\mathbf{H}'_i$  and  $\mathbf{H}'_{i+1}$  are equivalently distributed. Indeed, in this second intermediate hybrid, at the  $j$ -th encryption oracle query we compute  $[y_j]_T \leftarrow (\mathbf{f}^\top + P'_{i+1}(j_{|i+1}))[\mathbf{u}_j] + [\mathbf{x}_j]^\top \mathbf{F}^\top [\mathbf{u}_j]$ . Moreover, we have that  $P'_{i+1}(j_{|i+1})\mathbf{u}_j = P'_i(j_{|i})\mathbf{u}_j$ , in fact:

$$\begin{aligned} P'_{i+1}(j_{|i+1})\mathbf{u}_j &= \left( P_i^{(1-j[i+1])}(j_{|i})\mathbf{D}_{1-j[i+1]}^\perp + \tilde{P}_i^{(j[i+1])}(j_{|i})\mathbf{D}_{j[i+1]}^\perp \right) \mathbf{D}_{j[i+1]}\mathbf{r}_j \\ &= \left( P_i^{(1-j[i+1])}(j_{|i})\mathbf{D}_{1-j[i+1]}^\perp \right) \mathbf{D}_{j[i+1]}\mathbf{r}_j \\ &= \left( P_i^{(0)}(j_{|i})\mathbf{D}_0^\perp + P_i^{(1)}(j_{|i})\mathbf{D}_1^\perp \right) \mathbf{D}_{j[i+1]}\mathbf{r}_j \\ &= P'_i(j_{|i})\mathbf{D}_{j[i+1]}\mathbf{r}_j = P'_i(j_{|i})\mathbf{u}_j \end{aligned}$$

In the above derivation, we first applied the definitions of  $P_{i+1}$  and  $\mathbf{u}_j$ , then we simplified the second term by noticing that  $\mathbf{D}_{j[i+1]}^\perp \mathbf{D}_{j[i+1]} = 0$ , then for the same exact reason we can add the component  $P_i^{(j[i+1])}(j_{|i})\mathbf{D}_{j[i+1]}^\perp$ , and finally we have the definition of  $P'_i$ .

Similarly, for the decryption oracle queries with input  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$  where  $\exists d : \mathbf{u} \in \text{span}(\mathbf{D}_d)$ , we have that  $P'_{i+1}(j_{|i+1})\mathbf{u} = P'_i(j_{|i})\mathbf{u}$ . The derivation is identical to before. Thus the two intermediate hybrids are equivalent.

Finally, we show that the second intermediate hybrid,  $\mathbf{H}'_{i+1}$ , is statistically close to  $\mathbf{H}_{1,i,3}$ ; in fact, the only difference is that in the latter hybrid we use the function  $P_{j+1}(\cdot)\mathbf{D}^\perp$ . Equivalently as before, the two random functions are not equivalently distributed only when  $\text{span}(\bar{\mathbf{D}}_0 \parallel \bar{\mathbf{D}}_1)$  has rank less than  $2d$ , which happens with probability at most  $1/q$ . Thus  $|\epsilon_{\mathbf{H}_{1,i,2}} - \epsilon_{\mathbf{H}_{1,i,3}}| \leq \frac{2}{q}$ .

**Hybrid  $\mathbf{H}_{1,i,4}$ .** We remove the direct check  $[\mathbf{u}]_1 \in \text{span}([\mathbf{D}_1]_1) \cup \text{span}([\mathbf{D}_0]_1)$  introduced in  $\mathbf{H}_{1,i,2}$ . This removal can only increase the winning probability of the adversary. Thus  $\epsilon_{\mathbf{H}_{1,i,3}} \leq \epsilon_{\mathbf{H}_{1,i,4}}$ .

**Hybrid  $\mathbf{H}_{1,i,5}$ .** To decrypt, we increase the number of keys used by the decryption oracle to compute the bit  $b_1$ .

$$\begin{aligned} \mathcal{S} &:= \{j_i \| b : b \in \{0, 1\}\} && \text{if } \exists j \leq \text{ctr} : \mathbf{D}^{*\perp}[\mathbf{x}] = \mathbf{D}^{*\perp}[\mathbf{x}_j] \\ \mathcal{S} &:= \{j_i \| b : b \in \{0, 1\}, j \leq \text{ctr}\} && \text{otherwise} \end{aligned}$$

This change can only increase the winning probability of the adversary since the set of the strings  $\mathcal{S}$  used in  $\mathbf{H}_{1,i,5}$  contains the set of strings used in  $\mathbf{H}_{1,i,4}$ .

As for non-critical queries, we need to show that the view of the adversary does not change: in particular, any non-critical query that decrypts to  $\perp$  in  $\mathbf{H}_{1,i,4}$  should decrypt to  $\perp$  in  $\mathbf{H}_{1,i,5}$  as well. This is easy to prove when the decryption query has  $[\mathbf{u}] \in \text{span}([\mathbf{D}])$ : indeed, even if we modify the set  $\mathcal{S}$ , this change does not affect the way we decrypt such queries (recall that any key  $P_{i+1}(\cdot)$  is then multiplied by  $\mathbf{D}^\perp$ .) Also, a non-critical query could be a query for which it holds that there exists  $j \in [Q_{\text{Enc}}]$  such that  $\mathbf{D}^{*\perp}\mathbf{x}_j$  is equal to  $\mathbf{D}^{*\perp}\mathbf{x}$ . If a query of this form successfully decrypts in  $\mathbf{H}_{1,i,4}$ , the same happens in  $\mathbf{H}_{1,i,5}$ : again, this is because  $\mathcal{S}$  in the latter hybrid is a superset of  $\mathcal{S}$  in  $\mathbf{H}_{1,i,4}$ . But, it is still possible that a query of this form decrypts to  $\perp$  in  $\mathbf{H}_{1,i,4}$ , but the ‘augmented’  $\mathcal{S}$  in this new hybrid makes the consistency bit  $b_1$  be 1, for some new key: we bound the probability of a similar event since we know that the only way to learn the image of the random function  $P_{i+1}(\cdot)$  is via oracle queries to  $\mathcal{O}_{\text{dec}}$  and  $\mathcal{O}_{\text{enc}}$ . By union bound, we obtain a statistical distance of  $O(Q_{\text{Enc}}Q_{\text{Dec}}/q)$ .

$$\epsilon_{\mathbf{H}_{1,i,4}} - O(Q_{\text{Enc}}Q_{\text{Dec}}/q) \leq \epsilon_{\mathbf{H}_{1,i,5}}.$$

**Hybrid  $\mathbf{H}_{1,i,6}$ .** This hybrid is equivalent to the previous one, but the decryption oracle computes a different set  $\mathcal{S}$ , as follows:

$$\begin{aligned} \mathcal{S} &:= \{j_{i+1}\} && \text{if } \exists j \leq \text{ctr} : \mathbf{D}^{*\perp}[\mathbf{x}] = \mathbf{D}^{*\perp}[\mathbf{x}_j] \\ \mathcal{S} &:= \{j_{i+1} : j \leq \text{ctr}\} && \text{otherwise} \end{aligned}$$

Notice that the set  $\mathcal{S}$  as defined in  $\mathbf{H}_{1,i,6}$  might be a (strict) subset of the set  $\mathcal{S}$  as defined in  $\mathbf{H}_{1,i,5}$ . Thus the distinguishing event is that the consistency check would pass in  $\mathbf{H}_{1,i,5}$  but it would not pass in  $\mathbf{H}_{1,i,6}$ . In particular, such consistency check passes for an index of the form  $j_i \| 1$ , such that  $j[i+1] = 0$  and  $j \leq \text{ctr}$ , and by the definition of the distinguishing event the integer representation of  $(j_i \| 1) \cdot 2^{|\log Q_{\text{Enc}}| - i - 1}$  is bigger than  $\text{ctr}$ . Thus the key  $\mathbf{f}^\top + P_i(j_i \| 1)\mathbf{D}^\perp$  was never used for an encryption query. The only way an adversary can learn information about one of such keys is via decryption queries. In particular, each decryption query can at most decrease the set of possibilities (namely a valid  $y$  that matches the consistency check) by one. Moreover, the number of such keys is (very loosely) upper-bounded by  $Q_{\text{Enc}}$ , thus by union bound over all such keys and over all the decryption queries we obtain:  $|\epsilon_{\mathbf{H}_{1,i,6}} - \epsilon_{\mathbf{H}_{1,i,5}}| \leq \frac{Q_{\text{Enc}} \cdot Q_{\text{Dec}}}{q - Q_{\text{Dec}}}$ .

**Hybrid  $\mathbf{H}_{1,i+1,0}$ .** We then switch back the distribution of  $[\mathbf{u}_j]$  to the span of  $[\mathbf{D}_0]$ . This transition is the reverse of what we have done to move from  $\mathbf{H}_{1,i,0}$  to  $\mathbf{H}_{1,i,1}$ . We proceed in two steps:

- We first switch the  $j$ -th vector  $[\mathbf{u}_j]$  computed by the encryption oracle to a vector in the span of  $[(\mathbf{D}|\mathbf{U})]$ , where  $\mathbf{U}$  is uniform over  $\mathbb{Z}_q^{n \times d}$ , if the  $(i + 1)$ -th bit of the binary representation of  $j$  is equal to 1.
- Then, we switch the  $j$ -th vector  $[\mathbf{u}_j]$  computed by the encryption oracle to a vector in the span of  $[\mathbf{D}_0]$ .

Altogether we obtain an adversary  $\mathcal{C}$  such that:

$$|\epsilon_{\mathbf{H}_{1,i+1,0}} - \epsilon_{\mathbf{H}_{1,i,6}}| \leq 2(n - d) \text{Adv}_{\mathbb{G}_1, \mathcal{D}_{n,d}, \mathcal{C}}^{\text{mddh}}(\lambda) + \frac{2}{q - 1}.$$

It is easy to see that  $\epsilon_{\mathbf{H}_2} = \epsilon_{\mathbf{H}_{1, \lceil \log Q_{\text{Enc}} \rceil, 6}}$ . Next, we prove that  $\epsilon_{\mathbf{H}_2} \leq \frac{O(n^2)Q_{\text{Enc}}Q_{\text{Dec}}}{q}$ . We reduce the adversary  $\mathcal{A}$  playing in  $\mathbf{H}_2$  to an (unbounded) adversary  $\mathcal{B}$  upon which we can invoke the Lemma 1. We say that  $\mathcal{B}$  *forged a valid tuple* if the output of  $\mathcal{B}$  matches the event described in the lemma. For any assignments of the vector  $\mathbf{a}$  and of the matrix  $\mathbf{D}$  in the support of  $\mathcal{D}_{n,d}$ , we can consider in the Lemma 1 the matrix  $\mathbf{E}$  to be set equal to  $\mathbf{D}^*$ .

*Claim.*  $\Pr[\mathbf{H}_2 = 1] \leq \frac{O(n^2)Q_{\text{Enc}}Q_{\text{Dec}}}{q}$ .

Let  $(\mathbf{D}, \mathbf{D}^*, \mathbf{D}^\top \mathbf{f}, \mathbf{D}^\top \mathbf{F}, \mathbf{F}\mathbf{D}^*)$  be the tuple received by  $\mathcal{B}$  from the challenger. The adversary  $\mathcal{B}$  samples uniformly random values  $(\bar{\mathbf{f}}, \bar{\mathbf{F}})$  such that  $\bar{\mathbf{f}}^\top \mathbf{D} = \mathbf{f}^\top \mathbf{D}$ ,  $\bar{\mathbf{F}}^\top \mathbf{D} = \mathbf{F}^\top \mathbf{D}$  and  $\bar{\mathbf{F}}\mathbf{D}^* = \mathbf{F}\mathbf{D}^*$ . We can think of the tuple  $(\bar{\mathbf{f}}, \bar{\mathbf{F}})$  as a “fake” proving key that matches the verification key given by the challenger. Given  $\mathbf{D}$  and  $\mathbf{a}$ , the reduction  $\mathcal{B}$  can sample all the secret material needed to simulate the hybrid  $\mathbf{H}_2$ . In particular, it can compute the proving key and verification key of the proof system  $\mathbf{PS}$  and sample the challenge bit. The reduction  $\mathcal{B}$  samples an index value  $j_{\text{Enc}}^* \in [Q_{\text{Enc}}]$  and an index  $j_{\text{Dec}}^* \in [Q_{\text{Dec}}]$ . (Recall that  $Q_{\text{Enc}}$  and  $Q_{\text{Dec}}$  denote the number of encryption, resp. decryption queries made by  $\mathcal{A}$ .) At the  $j$ -th query to the encryption oracle:

- If  $j \neq j_{\text{Enc}}^*$ , the reduction  $\mathcal{B}$  generates  $\mathbf{x}_j$  following the prescribed algorithms. Then, it computes  $y_j \leftarrow \left( (\bar{\mathbf{f}} + \bar{\mathbf{F}}\mathbf{x}_j)^\top + P(j)\mathbf{D}^\perp \right) \mathbf{u}_j$ , where we recall that  $P(\cdot)$  is a random function.
- Else, for  $j = j_{\text{Enc}}^*$ ,  $\mathcal{B}$  computes  $\mathbf{x}_j$  as prescribed, queries its own oracle with  $\mathbf{x}_j$  and obtains a value  $\mathbf{v} = \mathbf{f} + \mathbf{F} \cdot \mathbf{x}_j$ , then, it uses  $\mathbf{v} + P(j)\mathbf{D}^\perp$  to compute the proof  $y$ , associated with  $\mathbf{u}_j$ , namely:  $y_j \leftarrow \left( \mathbf{v}^\top + P(j)\mathbf{D}^\perp \right) \mathbf{u}_j$ .

At the  $j$ -th query to decryption oracle with ciphertext  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$  there are three possible cases. The easiest case to handle is if  $\mathbf{u} \in \text{span}(\mathbf{D})$  or  $\exists j \neq j_{\text{Enc}}^*$  such that  $\mathbf{D}^{*\perp} \mathbf{x}_j = \mathbf{D}^{*\perp} \mathbf{x}$ . The reduction  $\mathcal{B}$  can compute the consistency check using the keys  $\bar{\mathbf{f}}, \bar{\mathbf{F}}$  and the random function  $P$ .



The second case is when  $\mathbf{D}^{*\perp} \mathbf{x}_{j_{\text{Enc}}^*} = \mathbf{D}^{*\perp} \mathbf{x}$ , in this case let  $\mathbf{r}'$  be such that  $\mathbf{x} - \mathbf{x}_{j_{\text{Enc}}^*} = \mathbf{D}^* \mathbf{r}'$  and compute

$$y' \leftarrow y_{j_{\text{Enc}}^*} + \mathbf{f}^\top \mathbf{D} \mathbf{r}' + \mathbf{x}_{j_{\text{Enc}}^*}^\top \mathbf{F}^\top \mathbf{D} \mathbf{r}' + (\mathbf{F} \mathbf{D}^* \mathbf{r}')^\top (\mathbf{u}_{j_{\text{Enc}}^*} + \mathbf{D} \mathbf{r}')$$

namely, compute the element  $[y']_T$  as if it was computed in the re-randomization of the ciphertext  $\mathbf{C}_{j_{\text{Enc}}^*}$  using randomness  $\mathbf{r}'$ . Notice that, by definition of  $\mathbf{H}_2$  the consistency check for  $[y]_T$  would be computed by checking if

$$y \stackrel{?}{=} \left( (\mathbf{f} + \mathbf{F} \mathbf{x})^\top + P(j_{\text{Enc}}^*) \mathbf{D}^\perp \right) \mathbf{u}.$$

By Lemma 2 and by definition of  $j_{j_{\text{Enc}}^*}$ , the two checks are equivalent. The last case is when  $\mathbf{u} \notin \text{span}(\mathbf{D}) \wedge \forall j : \mathbf{D}^{*\perp} \mathbf{x}_j \neq \mathbf{D}^{*\perp} \mathbf{x}$ , i.e., the query might be “critical”:

- If  $j < j_{\text{Dec}}^*$  then return  $\perp$  to the adversary  $\mathcal{A}$ , in this case we assume that the query was not critical and that the decryption would fail.
- If  $j = j_{\text{Dec}}^*$  then output the tuple  $(y - P(j_{\text{Enc}}^*) \mathbf{D}^\perp \mathbf{u}, \mathbf{u}, \mathbf{x})$  as the forgery of  $\mathcal{B}$ .

We condition on the event that  $j_{\text{Dec}}^*$  is the first critical query of  $\mathcal{A}$  and that, let the ciphertext sent by  $\mathcal{A}$  at the  $j_{\text{Dec}}^*$  query be  $\mathbf{C} = ([\mathbf{x}], [y]_T, \pi)$  we have that the equation  $[y]_T = (\mathbf{f} + P(j_{\text{Enc}}^*) \mathbf{D}^\perp + \mathbf{F} \mathbf{x})^\top [\mathbf{u}]$  holds. Let **Guess** be such event. Conditioned on such a lucky event,  $\mathcal{B}$  indeed produces a valid forgery, in fact by the definition of a critical query  $(\mathbf{x}_{j_{\text{Enc}}^*} - \mathbf{x}) \notin \text{span}(\mathbf{D}^*)$  and  $\mathbf{u} \notin \text{span}(\mathbf{D})$ .

We show that the view provided by  $\mathcal{B}$  to the adversary  $\mathcal{A}$  up to the  $j_{\text{Dec}}^*$ -th decryption query and conditioned on **Guess** is equivalent to the view of the adversary up to the  $j_{\text{Dec}}^*$ -th decryption query in the hybrid game  $\mathbf{H}_2$ . The intuition is that the values  $P(j) \mathbf{D}^\perp$ , for all  $j$ , mask the components of  $(\mathbf{f}, \mathbf{F})$  and  $(\bar{\mathbf{f}}, \bar{\mathbf{F}})$  that differ. Indeed, we know that for some row vectors  $\mathbf{v}, \mathbf{w}, \mathbf{w}'$ , it holds that  $\mathbf{f} = \mathbf{D} \mathbf{v} + (\mathbf{w} \mathbf{D}^\perp)^\top$  and  $\bar{\mathbf{f}} = \mathbf{D} \mathbf{v} + (\mathbf{w}' \mathbf{D}^\perp)^\top$ . Similarly, for some  $\mathbf{V}, \mathbf{W}$  and  $\mathbf{W}'$ ,  $\mathbf{F} = \mathbf{D} \mathbf{V} + (\mathbf{W} \mathbf{D}^\perp)^\top$ , and  $\bar{\mathbf{F}} = \mathbf{D} \mathbf{V} + (\mathbf{W}' \mathbf{D}^\perp)^\top$ .

Let  $P'$  be a uniformly random function, and consider the following function:

$$P(j) = \begin{cases} P'(j), & j = j_{\text{Enc}}^* \\ P'(j) + \Delta_j, & j \neq j_{\text{Enc}}^* \end{cases}$$

where  $\Delta_j = \mathbf{w} - \mathbf{w}' + \mathbf{x}_j^\top (\mathbf{W} - \mathbf{W}')$ . It is not hard to see that  $P$  is a uniformly random function. Now consider the mental experiment where  $\mathcal{B}$  runs the same but using the random function  $P$  defined above. Since  $P$  is uniformly random, the probability that  $\mathcal{B}$  forges a valid tuple in this mental experiment is the same as the probability that  $\mathcal{B}$  forges a valid tuple in the real experiment. Also, for any  $j \neq j_{\text{Enc}}^*$  the value  $y$  computed at the  $j$ -th encryption oracle query is:

$$\begin{aligned} y &= \left( (\bar{\mathbf{f}} + \bar{\mathbf{F}} \mathbf{x}_j)^\top + P(j) \mathbf{D}^\perp \right) [\mathbf{u}_j] \left( (\bar{\mathbf{f}} + \bar{\mathbf{F}} \mathbf{x}_j)^\top + (P'(j) + \Delta_j) \mathbf{D}^\perp \right) [\mathbf{u}_j] \\ &= \left( \left( \bar{\mathbf{f}} + ((\mathbf{w} - \mathbf{w}') \mathbf{D}^\perp)^\top + (\bar{\mathbf{F}} + ((\mathbf{W} - \mathbf{W}') \mathbf{D}^\perp)^\top) \mathbf{x}_j \right)^\top + P'(j) \mathbf{D}^\perp \right) [\mathbf{u}_j] \\ &= \left( (\mathbf{f} + \mathbf{F} \mathbf{x}_j)^\top + P'(j) \mathbf{D}^\perp \right) [\mathbf{u}_j]. \end{aligned}$$

The probability that the reduction  $\mathcal{B}$  creates a forgery is  $\Pr[\mathbf{H}_2 = 1 \wedge \mathbf{Guess}]$ , and the two events are independent. Moreover, since  $\Pr[\mathbf{Guess}] = (Q_{\text{Enc}}Q_{\text{Dec}})^{-1}$ , by the Rand-RCCA Lemma in [14] we have that  $\Pr[\mathbf{H}_2 = 1] \leq \frac{n(n+1)Q_{\text{Enc}}Q_{\text{Dec}}}{q}$ .

### 5.1 Publicly-Verifiable Rand-RCCA PKE

We show two publicly verifiable Rand-RCCA PKE schemes based on the scheme from Sect. 5. Following the ideas in [13], we append a malleable NIZK proof (essentially a Groth-Sahai proof) that  $[y]_T$  and  $\pi$  are well-formed to the ciphertexts of  $\mathcal{PK}\mathcal{E}$  from the previous section. The decryption algorithm outputs the decrypted message only if the NIZK proofs are valid. Public verifiability follows because the NIZK proofs can be verified using the public parameters.

Let  $\mathcal{PK}\mathcal{E}_1 = (\text{KGen}_1, \text{Enc}_1, \text{Dec}_1, \text{Rand}_1)$  be the scheme of Sect. 5 instantiated using the benign proof system of Sect. 3.1, and let  $\mathbf{PS}_2 = (\text{PGen}_2, \text{PPrv}_2, \text{PVer}_2)$  and  $\text{PEV}_2$  form a malleable NIZK system for membership in the relation

$$\mathcal{R}_2 = \left\{ (\text{pk}, [\mathbf{x}]), ([y]_T, \pi, \mathbf{r}) : \begin{array}{l} y = \mathbf{f}^\top \mathbf{u} + \mathbf{x}^\top \mathbf{F} \mathbf{u} \\ \text{PPrv}_1(ppk, [\mathbf{u}], \mathbf{r}) = \pi \end{array} \right\},$$

and where the allowable set of transformations contains all the transformations  $(T_{\text{el}}, T_{\text{wit}})$  such that it exists  $\hat{\mathbf{r}}$  with  $T_{\text{el}}(\text{pk}, [\mathbf{x}]) = \text{pk}, [\hat{\mathbf{x}}]$ ,  $T_{\text{wit}}([y]_T, \pi, \mathbf{r}) = [\hat{y}]_T, \hat{\text{pk}}, \mathbf{r} + \hat{\mathbf{r}}$  and  $([\hat{\mathbf{x}}], [\hat{y}]_T, \hat{\pi}) = \text{Rand}_1(\text{pk}, ([\mathbf{x}], [y]_T, \pi); \hat{\mathbf{r}})$ ; each transformation in the set of allowable transformation is uniquely identified by a vector  $\hat{\mathbf{r}}$ .

The pv-Rand-PKE scheme  $\mathcal{PK}\mathcal{E}_2 = (\text{Init}, \text{KGen}_2, \text{Enc}_2, \text{Dec}_2, \text{Rand}_2, \text{Ver})$  is identical to  $\mathcal{PK}\mathcal{E}_1$ , except that (i)  $\text{KGen}_2$  additionally samples the common reference string for  $\mathbf{PS}_2$ , (ii) the encryption procedure computes a ciphertext as in  $\mathcal{PK}\mathcal{E}_1$  but additionally computes a proof  $\pi_2$  for  $\mathbf{PS}_2$  and outputs a ciphertext  $\mathbf{C} = ([\mathbf{x}_1], \pi_2)$ , (iii) the decryption procedure first checks the proof  $\pi_2$  holds w.r.t. the instance  $(\text{pk}, [\mathbf{x}])$  and, if so, it outputs  $\mathbf{M} = (-\mathbf{a}^\top, 1)[\mathbf{x}]$  (and  $\perp$  otherwise), (iv) the re-randomization procedure randomizes  $[\mathbf{x}]$  as in  $\mathcal{PK}\mathcal{E}_1$  and uses  $\text{PEV}_2$  for the remaining part of the ciphertext, and (v)  $\text{Ver}_2$  simply checks  $\pi_2$ .

**Theorem 4.** *If  $\mathbf{PS}_2$  is adaptively sound,  $(\epsilon, O(T))$ -composable zero-knowledge, and perfect derivation private, and  $\mathcal{PK}\mathcal{E}_1$  is mRCCA secure then  $\mathcal{PK}\mathcal{E}_2$  is publicly verifiable, perfectly re-randomizable, and mRCCA-secure. Specifically, for any PPT  $\mathcal{A}$  making up to  $Q_{\text{Enc}}$  encryption queries and  $Q_{\text{Dec}}$  decryption queries and with running time  $T$  exist PPT  $\mathcal{B}^{\text{rcca}}$  making the same number of queries and adversaries  $\mathcal{B}^{\text{snd}}, \mathcal{B}^{\text{zk}}$  with similar running times*

$$\text{Adv}_{\mathcal{A}, \mathcal{PK}\mathcal{E}_2}^{\text{mRCCA}}(\lambda) \leq \text{Adv}_{\mathcal{B}^{\text{rcca}}, \mathcal{PK}\mathcal{E}_1}^{\text{mRCCA}}(\lambda) + \text{Adv}_{\mathcal{B}^{\text{snd}}, \mathbf{PS}_2}^{\text{snd}}(\lambda) + \epsilon$$

The proof follows by inspection of the proof of Theorem 2 in [13]. In more detail, their proof proceeds in two steps. First, it reduces to the adaptive soundness of the NIZK proof system to claim that if a *publicly-verifiable* ciphertext decrypts correctly then its respective *non-publicly verifiable* ciphertext should decrypt correctly too. We notice that this step can be performed tightly relying either

on statistical adaptive soundness of the proof system or relying on the computational soundness of the proof system when the language proved is witness samplable. The reason is that the reduction can check which one of the many NIZK-proofs from the adversary breaks adaptive soundness before submitting it as its forgery. The second step uses composable zero-knowledge to first tightly switch the way the public parameters are generated and then to switch (all together) the proofs for the ciphertexts from real to simulated.

To instantiate the malleable NIZK, we consider a construction along the same line of [13]. In more detail, [13] introduced an extension of the Groth-Sahai proof system that is zero-knowledge even for pairing product equations where the  $\mathbb{G}_T$ -elements are variables. Their idea is to commit the elements in  $\mathbb{G}_T$  using a commitment scheme with nice bilinear properties. Groth-Sahai proofs can be instantiated under any  $\mathcal{D}_k$ -MDDH Assumption [11] and, given their nice algebraic properties they are malleable [8]. More details are given in [14].

**A More Efficient Tight-Secure pv-Rand-RCCA PKE.** To facilitate our more efficient scheme, we introduce a stronger variant of the MDDH assumption (cf. Definition 1) in which the adversary gets not only a matrix  $[\mathbf{A}]$ , but also the tensor product  $[\mathbf{A} \otimes \mathbf{A}]$  to distinguish an element from  $span([\mathbf{A}])$  and random:

**Definition 9 (Tensor Matrix Diffie-Hellman assumption in  $\mathbb{G}_\gamma$ ).** *The  $\mathcal{D}_{\ell,k}$ -Tensor-Matrix-Decisional-Diffie-Hellman (TMDDH) assumption in group  $\mathbb{G}_\gamma$  holds if for all non-uniform PPT adversaries  $\mathcal{A}$ ,*

$$|\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A} \otimes \mathbf{A}]_\gamma, [\mathbf{A}]_\gamma, [\mathbf{A}\mathbf{w}]_\gamma) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A} \otimes \mathbf{A}]_\gamma, [\mathbf{A}]_\gamma, [\mathbf{z}]_\gamma) = 1]|$$

*is negligible, where the probability is taken over  $\mathcal{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2) \leftarrow \text{GGen}(1^\lambda)$ ,  $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$ ,  $\mathbf{w} \leftarrow \mathbb{Z}_q^k$ ,  $[\mathbf{z}]_\gamma \leftarrow \mathbb{G}_\gamma^\ell$ , and the coin tosses of adversary  $\mathcal{A}$ .*

The TMDDH assumption can be seen as a generalization of the “square-Diffie-Hellman” assumption [6, 29], and as a special case of the “Uber assumption family” [5]. Since a TMDDH adversary gets quadratic terms  $[\mathbf{A} \otimes \mathbf{A}]$  “in the exponent”, it is not clear how this assumption relates to the more standard MDDH assumption. However, we remark that the TMDDH assumption holds generically for large enough dimensions, at least for uniformly random  $\mathbf{A}$ .

**Lemma 4 (Generic security of TMDDH).** *For  $k \geq 4$ , the  $\mathcal{U}_{k+1,k}$ -TMDDH assumption holds against generic adversaries in a symmetric pairing setting.*

In [14] we explain what we mean by “holds generically” according to the formulation of Maurer [28] and we sketch a proof of the lemma.

The idea of the second publicly-verifiable PKE scheme is to (1) add in the public key the values  $\mathbf{k}^\top [\mathbf{D} \otimes \mathbf{D}]$  and (2) use a malleable proof system  $\mathbf{PS}_3$  for membership in the relation

$$\mathcal{R}_3 = \left\{ (\text{pk}, [\mathbf{x}]), ([y]_T, \pi, \mathbf{r}) : \begin{array}{l} y = \mathbf{f}^\top \mathbf{u} + \mathbf{x}^\top \mathbf{F} \mathbf{u} \\ \mathbf{k}^\top [\mathbf{D} \otimes \mathbf{D}] \mathbf{r} \otimes \mathbf{r} \cdot [1] = \pi \end{array} \right\},$$

with the same set of allowable transformations as in the previous publicly verifiable PKE scheme. The languages associated with the relation  $\mathcal{R}_3$  and  $\mathcal{R}_2$  are identical, but we can obtain a more efficient NIZK proof for  $\mathcal{R}_3$ .

**Theorem 5.** *The pv-Rand-PKE scheme  $\mathcal{PK}\mathcal{E}_3$  is publicly verifiable, perfectly re-randomizable and RCCA-secure. Specifically:*

$$\begin{aligned} \mathbf{Adv}_{\mathcal{A}, \mathcal{PK}\mathcal{E}}^{\text{RCCA}}(\lambda) \leq & \mathbf{Adv}_{\mathbb{G}_1, \mathcal{U}_{n,d}, \mathcal{B}}^{\text{TMDDH}}(\lambda) + O(d \log Q_{\text{Enc}}) \cdot \mathbf{Adv}_{\mathbb{G}_1, \mathcal{U}_{n,d}, \mathcal{B}'}^{\text{MDDH}}(\lambda) \\ & + \log Q_{\text{Enc}} \cdot \mathbf{Adv}_{\mathcal{B}'', \mathbf{PS}}^{\text{snd}}(\lambda) + O\left(\frac{n^2 Q_{\text{Dec}} Q_{\text{Enc}} \log Q_{\text{Enc}}}{q}\right) \end{aligned}$$

We only sketch the proof, which is only a slight variation of the proof of Theorem 3. Notice that in the proof of Theorem 3 to move from  $\mathbf{G}_3$  to  $\mathbf{G}_4$  we use the  $\mathcal{D}_{n,d}$ -MDDH assumption. This step changes with our modified scheme, since we add  $[\mathbf{D} \otimes \mathbf{D}]$  to the public key. We thus need to rely on the stronger TMDDH assumption. Also notice that this is the only step in the proof of Theorem 3 where the assumption over the matrix  $[\mathbf{D}]$  is used. Finally, observe that we can prove both composable zero-knowledge and computational adaptive soundness of the NIZK proof system for  $\mathcal{R}_3$  using the classical  $\mathcal{D}_k$ -MDDH assumption.

## 6 Application: Universally Composable MixNet

We can plug-and-play our pv-Rand-RCCA PKE schemes in the MixNet protocol of [13] because their protocol works for any pv-Rand RCCA scheme that has the property of being *linear* and a property that holds for both  $\mathcal{PK}\mathcal{E}_2$  and  $\mathcal{PK}\mathcal{E}_3$ . For space reasons, we defer the details in [14].

The MixNet ideal functionality interacts with  $n$  sender parties and  $m$  mixer parties. The  $i$ -th sender sends the message  $M_i$ , while the mixer can decide to mix the messages. At the end, when all the mixer have sent their inputs, the functionality returns the list of sorted messages. For space reasons, the ideal functionality is formally defined in [14].

The protocol is divided into 3 phases: (i) at the input phase, the *sender parties* send pv-Rand-RCCA ciphertexts of their messages and a simulation-extractable<sup>5</sup> NIZK of knowledge; (ii) at the mixing phase, the mixers, one after the other, shuffle the ciphertexts and compute the so-called *check-sum* NIZK proofs that paired with the public-verifiability and the RCCA property are sufficient to prove the validity of the shuffles; (iii) at the output phase, the ciphertexts are decrypted. The nice feature of the protocol is that the statements proved by the *check-sum* proofs are of constant size, independent of the number of shuffled ciphertexts.

The NIZK proofs employed in the input-submission phase are needed only to make sure independence of the inputs. We notice that to obtain our “tightly-secure” MixNet we need only to make sure that the Rand-mRCCA PKE and the simulation-extractable NIZK proofs are tightly secure. Let  $\mathbf{Adv}_{\mathcal{A}, \mathbf{PS}}^{\text{sim-ext}}(\lambda)$  be the advantage of an adversary  $\mathcal{A}$  against the simulation extractability experiment for  $\mathbf{PS}$ , we are ready now to state the main contribution of this section.

<sup>5</sup> Actually, they need a weaker form of soundness called all-but-one soundness, however simulation extractability is sufficient.

**Theorem 6.** Let  $\mathcal{PK}\mathcal{E}$  be a linear *pv-Rand* *RCCA* *PKE*,  $\mathbf{PS}$  be a simulation-extractable *NIZK*, and let  $\Pi$  be the *MixNet* protocol from [13] instantiated with  $\mathcal{PK}\mathcal{E}$  and  $\mathbf{PS}$ . The protocol  $\Pi$  realizes  $\mathcal{F}_{\text{Mix}}$  with setup assumptions a threshold decryption functionality  $\mathcal{F}_{\text{TDdec}}[\mathcal{PK}\mathcal{E}]$  and a common-reference string functionality  $\mathcal{F}_{\text{CRS}}$ . More in detail, there exist a simulator  $\mathcal{S}$  and negligible function  $\text{negl}(\lambda, m)$  such that for any static-corruption environment  $\mathcal{Z}$  with running time  $T_{\mathcal{Z}}$  there exist an adversaries  $\mathcal{B}, \mathcal{B}'$  whose running time is  $O(T_{\mathcal{Z}}(\lambda))$ , such that:

$$\begin{aligned} & \left| \Pr[\text{REAL}_{\mathcal{Z}, \Pi}(\lambda) = 1] - \Pr[\{\mathcal{F}_{\text{CRS}}, \mathcal{F}_{\text{TDdec}}\}\text{-HYBRID}_{\mathcal{Z}, \mathcal{S}}^{\mathcal{F}_{\text{Mix}}}(\lambda) = 1] \right| \\ & \leq 3\text{Adv}_{\mathcal{B}, \mathcal{PK}\mathcal{E}}^{\text{mRCCA}}(\lambda) + \text{Adv}_{\mathcal{B}', \mathbf{PS}'}^{\text{sim-ext}}(\lambda) + \text{negl}(\lambda, m) \end{aligned}$$

We stress that the function  $\text{negl}(\lambda, m)$  in the statement of Theorem 6 is independent of  $T_{\mathcal{Z}}$  and only depends on the number of mixers (which we can think as a small number). The proof of the theorem follows by inspection of the proof of Theorem 5 in [13] and observing that the three steps of the proof that reduce to the *pv-Rand-RCCA* security of  $\mathcal{PK}\mathcal{E}$  can be performed tightly by relying on the multi-ciphertext *RCCA* security definition (cf. Definition 8). In [14] we give more details and we show how to instantiate the necessary simulation-extractable *NIZK* using the tightly-secure *QA-NIZK* based on the *MDDH* assumption of Abe *et al.* [2]. Thus, instantiating the protocol with  $\mathcal{PK}\mathcal{E}_2$  (resp.  $\mathcal{PK}\mathcal{E}_3$ ) we obtain a *MixNet* protocol that reduces almost-tightly in the number of mixed messages to the *MDDH* (resp. *TMDDH*) Assumption.

## References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: new constructions and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 69–100. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46803-6\\_3](https://doi.org/10.1007/978-3-662-46803-6_3)
2. Abe, M., Jutla, C.S., Ohkubo, M., Roy, A.: Improved (Almost) tightly-secure simulation-sound *QA-NIZK* with applications. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11272, pp. 627–656. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03326-2\\_21](https://doi.org/10.1007/978-3-030-03326-2_21)
3. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_17](https://doi.org/10.1007/978-3-642-29011-4_17)
4. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45539-6\\_18](https://doi.org/10.1007/3-540-45539-6_18)
5. Boyen, X.: The uber-assumption family (invited talk). In: Pairing 2008 (2008)
6. Burmester, M., Desmedt, Y., Seberry, J.: Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically) extended abstract. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 380–391. Springer, Heidelberg (1998). [https://doi.org/10.1007/3-540-49649-1\\_30](https://doi.org/10.1007/3-540-49649-1_30)

7. 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](https://doi.org/10.1007/978-3-540-45146-4_33)
8. 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](https://doi.org/10.1007/978-3-642-29011-4_18)
9. 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>
10. 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. LNCS, vol. 9814, pp. 341–372. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53018-4\\_13](https://doi.org/10.1007/978-3-662-53018-4_13)
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](https://doi.org/10.1007/978-3-642-40084-1_8)
12. 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](https://doi.org/10.1007/978-3-030-57808-4_14)
13. 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](https://doi.org/10.1007/978-3-030-34618-8_6)
14. Faonio, A., Hofheinz, D., Russo, L.: Almost tightly-secure re-randomizable and replayable CCA-secure public key encryption. Cryptology ePrint Archive, Paper 2023/152 (2023). <https://eprint.iacr.org/2023/152>
15. Faonio, A., Russo, L.: Mix-nets from re-randomizable and replayable CCA-secure public-key encryption. In: Security and Cryptography for Networks (2022). [https://doi.org/10.1007/978-3-031-14791-3\\_8](https://doi.org/10.1007/978-3-031-14791-3_8)
16. Fauzi, P., Lipmaa, H., Siim, J., Zając, M.: An efficient pairing-based shuffle argument. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 97–127. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70697-9\\_4](https://doi.org/10.1007/978-3-319-70697-9_4)
17. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49890-3\\_1](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. LNCS, vol. 10403, pp. 133–160. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_5](https://doi.org/10.1007/978-3-319-63697-9_5)
19. Gay, R., Hofheinz, D., Kohl, L., Pan, J.: More efficient (almost) tightly secure structure-preserving signatures. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 230–258. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78375-8\\_8](https://doi.org/10.1007/978-3-319-78375-8_8)
20. 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](https://doi.org/10.1007/978-3-540-24638-1_9)

21. 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. LNCS, vol. 11693, pp. 417–447. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26951-7\\_15](https://doi.org/10.1007/978-3-030-26951-7_15)
22. Hofheinz, D.: Adaptive partitioning. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 489–518. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_17](https://doi.org/10.1007/978-3-319-56617-7_17)
23. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-42033-7\\_1](https://doi.org/10.1007/978-3-642-42033-7_1)
24. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_26](https://doi.org/10.1007/978-3-540-28628-8_26)
25. Libert, B., Joye, M., Yung, M., Peters, T.: Concise Multi-challenge CCA-secure encryption and signatures with almost tight security. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 1–21. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-45608-8\\_1](https://doi.org/10.1007/978-3-662-45608-8_1)
26. Libert, B., Peters, T., Joye, M., Yung, M.: Compactly hiding linear spans. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 681–707. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48797-6\\_28](https://doi.org/10.1007/978-3-662-48797-6_28)
27. Libert, B., Peters, T., Qian, C.: Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 247–276. Springer, Heidelberg (2017). [https://doi.org/10.1007/978-3-662-54365-8\\_11](https://doi.org/10.1007/978-3-662-54365-8_11)
28. Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: 10th IMA International Conference on Cryptography and Coding (2005)
29. Maurer, U.M., Wolf, S.: Diffie-hellman oracles. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 268–282. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-68697-5\\_21](https://doi.org/10.1007/3-540-68697-5_21)
30. Naveed, M., et al.: Controlled functional encryption. In: ACM CCS 2014 (2014)
31. 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](https://doi.org/10.1007/978-3-319-70278-0_22)
32. 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](https://doi.org/10.1007/978-3-540-74143-5_29)
33. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992). [https://doi.org/10.1007/3-540-46766-1\\_35](https://doi.org/10.1007/3-540-46766-1_35)
34. Tarjan, R.E.: Efficiency of a good but not linear set union algorithm. J. ACM (1975)
35. Wang, Y., Chen, R., Yang, G., Huang, X., Wang, B., Yung, M.: Receiver-anonymity in rerandomizable RCCA-secure cryptosystems resolved. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 270–300. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84259-8\\_10](https://doi.org/10.1007/978-3-030-84259-8_10)



# Multi-authority ABE for Non-monotonic Access Structures

Miguel Ambrona<sup>1</sup> and Romain Gay<sup>2</sup>(✉)

<sup>1</sup> Nomadic Labs, Paris, France

<sup>2</sup> IBM Research Zurich, Zurich, Switzerland

romain.rgay@gmail.com

**Abstract.** Attribute-Based Encryption (ABE) is a cryptographic primitive which supports fine-grained access control on encrypted data, making it an appealing building block for many applications. Multi-Authority Attribute-Based Encryption (MA-ABE) is a generalization of ABE where the central authority is distributed across several independent parties.

We provide the first MA-ABE scheme from asymmetric prime-order pairings where no trusted setup is needed and where the attribute universe of each authority is unbounded. Moreover, it is the first to handle non-monotonic access structures. These features broaden the applicability and improve the efficiency of our scheme. Our construction makes a modular use of Functional Encryption schemes with fine-grained access control.

## 1 Introduction

Attribute-Based Encryption (ABE) [SW05, GPSW06] subsumes traditional public-key encryption by providing fine-grained access to the encrypted data. Namely, each ciphertext is associated with an access policy, and each user receives a so-called user secret key according to their credentials. If these credentials fulfill the policy, the user secret key can be used to successfully decrypt the ciphertext. Otherwise, the plaintext remains hidden, even if several non-authorized users collude.

Despite being a prominent topic in the research community, the notion of ABE suffers from several drawbacks. User secret keys are generated from a so-called master secret key, which can decrypt any ciphertext. Consequently, the generation of these keys must be performed by a trusted third party, who controls the master secret key and who must be online every time a key is requested (not only during the setup phase of the scheme). Such a third party is a single point of failure in the system and is likely to be a target for attacks. Copying the master secret and using redundant servers to alleviate this bottleneck only increases the chances of key exposure. Besides, the master secret key owner can impersonate any user of its choice, acting as an escrow (see [Rog15] for further details on this issue). To mitigate these shortcomings, a solution is to decentralize the key-generation so that no single party holds the master secret key in full. Furthermore, decentralization is encouraged given that in many scenarios



the access policy used to generate a ciphertext includes attributes coming from different organizations.

The work of [Cha07] and later [MKE08] considered a variation of ABE where any party can become an authority by publishing some public key; these authorities, created on the fly, handle different attributes, and no coordination is required among them. In these systems, a user equipped with a global identifier can collect different credentials associated with different attributes from each authority. However, the user must then interact with a trusted central authority that will process such credentials and provide the actual ABE user secret keys. The advantage of their approach is that this central authority is agnostic to the meaning of the attributes and credentials of the user, and does not need to communicate with the other authorities. However, most of the aforementioned shortcomings remain. Afterward, [LCLS08] removed the need for a central authority, but the set of authorities in their construction is fixed and they must interact during the setup phase. Another limitation is that the security of their scheme is only proven for an a priori bounded number of collusions. [CC09] also presented a scheme with no central authority relying on distributed PRFs. However, their scheme is still limited in terms of expressiveness (it can only express a strict AND policy) and only handles a pre-determined set of attributes. In [LW11], the authors gave the first construction where there is no central authority, authorities can join the system on the fly without communicating with each other and the ciphertexts can be associated with a rich class of expressive access policies (including Boolean formulas). Despite these impressive features, their construction still suffers from some limitations: it requires a trusted setup; it uses inefficient composite-order pairings; each authority can only handle a small (poly-size) set of attributes as, in fact, the public key of each authority grows with the number of attributes owned by the authority. Later on, in [OT13, RW15], the authors built MA-ABE where there is no trusted setup beyond the mere agreement of which groups and which hash function to use, and where the attribute set of each authority is of exponential size or unbounded. Moreover, these schemes have the advantage of using prime-order pairings, which are more efficient than their composite-order counterparts. However, the scheme from [OT13] is not shown to achieve security in the presence of corrupted authorities, an important requirement in the standard security definition for MA-ABE. The scheme from [RW15] inherits from [LW11] prohibitively large ciphertexts. Indeed, in these schemes, each ciphertext contains a number of *target* group elements that grows with the size of its associated access policy, which are significantly larger than source group elements. Another reason all existing schemes lack practical efficiency is their use of *symmetric* pairings, which are less efficient than their asymmetric counterparts. This is in contrast with state-of-the-art single-authority ABE schemes, defined over *asymmetric* pairings and without target group elements in the ciphertext.

Finally, existing MA-ABE can only handle monotonic access structures. Namely, policies that can be expressed by a Boolean formula with *positive literals* only, e.g. of the form:  $\text{Role} = \text{Reviewer} \wedge \text{Year} = 2022$ . Suppose the document

to be encrypted is an audit of the security department of some company for the year 2022. In order to avoid conflict of interests, employees from the security department should not be able to access the document. This corresponds to the non-monotonic formula:  $(\text{Role} = \text{Reviewer} \wedge \text{Year} = 2022) \wedge \neg(\text{Department} = \text{Security})$ . A naive way to implement negative literals would be to give user secret keys associated with both the credential owned by the user and all the negative literals not owned by the user, e.g.  $\neg(\text{HumanResources})$ ,  $\neg(\text{IT})$ ,  $\neg(\text{Marketing})$ ,  $\neg(\text{R\&D})$ ,  $\neg(\text{Production})$ , and so on, for all existing departments in the company where the user does not belong. This solution yields very large user secret keys, since they grow proportionally with the number of possible attributes. In fact, this becomes unfeasible for large attribute universe (where the number of attribute is super-polynomial), let alone unbounded universe (where there is no restriction on the number of possible attributes, i.e. any bit string can serve as an attribute). [OSW07] gave the first ABE for non-monotonic formulas, but their techniques do not seem to be directly applicable to the multi-authority setting. This prompts the question: *Can we achieve MA-ABE with similar features and efficiency than single-authority ABE?*

*Our contribution.* We provide the first MA-ABE scheme from asymmetric prime-order pairings, without trusted setup and where the attribute universe of each authority is of unbounded size. Furthermore, our scheme handles non-monotonic access structures. It makes a modular use of practical Functional Encryption (FE) schemes for simple functions, namely, inner-products (we refer to our technical overview for more details about the FE we use). We prove security from standard assumptions using pairings (namely, the SXDH assumption) in the random oracle model. Our construction achieves security against adversaries that can choose the access structure of the challenge ciphertext and the attributes of the user secret keys, but the access structure and the attributes chosen cannot depend on the cryptographic material received. That is, they must not depend on the challenge ciphertext or the user secret keys (although they can depend on the public key). We refer to this security notion as super-selective security—the selective security notion traditionally refers to the setting where the adversary is constrained to choose the access structure used in the challenge ciphertext before receiving any cryptographic material (either the public key or the user secret keys). We leave it as an open problem to obtain adaptive security. Table 1 compares our scheme with the state-of-the-art.

*Technical overview.* We consider an MA-ABE where access policies are represented by monotone span programs (MSP) (as per Definition 1), which capture monotonic Boolean formulas. We explain how to handle non-monotonic formulas later in this overview. In a nutshell, an MSP allows users to produce shares  $s_1, \dots, s_\ell$  of a secret  $s$ , where  $\ell$  is the size of the MSP, and each share  $s_j$  is associated with an attribute  $\rho(j)$ . Akin to standard secret sharing schemes, the secret  $s$  can be recovered if and only if sufficiently many shares  $s_j$  are given. The ABE uses cyclic groups  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$  of prime order  $p$ , equipped with a bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ . We use additive bracket notation for all three groups,

**Table 1.** Comparison among MA-ABE schemes. The attribute universe is said to be small when it is a-priori bounded by a polynomial in the security parameter. It is said to be large when it is of a-priori bounded exponential size in the security parameters, or not bounded at all. “Corrupted authorities” refers to whether or not the scheme is secure when the adversary can acquire the secret keys of some authorities of their choice, or even create authorities with a public key of their choice (this is the standard definition for MA-ABE). “q-type” refers to a family of parameterized computational assumptions in pairing groups. “s-selective” refers to the super-selective security notion (defined in Sect. 2.5).

Reference	[LW11]	[RW15]	[OT13]	This work
pairing type	comp. sym.	prime. sym.	prime sym.	prime asym.
assumption	composite	q-type	DLIN	SXDH
security	adaptive	selective	adaptive	s-selective
attribute universe	small	large	large	large
attributes per authority	bounded	unbounded	bounded	unbounded
non-monotonic access structures	no	no	no	yes
corrupted authorities	yes	yes	no	yes

that is, for  $s \in \{1, 2, t\}$ , and all scalars  $x \in \mathbb{Z}_p$ , we write  $\llbracket x \rrbracket_s = xP_s$  where  $P_s$  is a generator of  $\mathbb{G}_s$ . Finally, we make use of Functional Encryption (FE) schemes, which are an advanced form of public-key encryption where the secret key can be used to derive functional secret keys  $\text{sk}_f$  for certain functions  $f$ . Decryption can use  $\text{sk}_f$  to extract from an encryption  $\text{Enc}(\text{pk}, m)$  of the message  $m$  the value  $f(m)$ . Nothing else is revealed about the message  $m$  apart from the value  $f(m)$ . Many functional secret keys can be derived for different functions from the secret key (which is referred to as master secret key, just like in the ABE setting). In short, FE enables selective computations on encrypted data. We rely on practical FE schemes that handle a particular class of functions of interest.

For encryption, an exponent  $s$  is uniformly sampled from  $\mathbb{Z}_p$  and the encapsulation key is defined as  $\llbracket s \rrbracket_t$  (we consider the KEM variant of ABE). The MSP is used to create shares  $\{s_j\}_{j \in [\ell]}$  of  $s$  and shares  $\{u_j\}_{j \in [\ell]}$  of 0. The MA-ABE ciphertext consists of one FE ciphertext of the vector  $(s_j, u_j)$  per  $j \in [\ell]$ . The public key of the FE used for each  $j \in [\ell]$  is published by the authority that owns the attribute  $\rho(j)$ . Note that in order to register into the system, each authority will run the FE setup algorithm to create its pair of keys  $(\text{FE.pk}, \text{FE.msk})$ .

The FE we are using is for identity-based inner-products. That is, each ciphertext encrypts a vector  $\mathbf{x}$  (of some fixed dimension, say  $d$ , which is then set to 2 for our modular construction), and an identity  $\text{id}$ . Each functional secret key is associated with a vector  $\llbracket \mathbf{y} \rrbracket_2 \in \mathbb{G}_2^d$  and an identity  $\text{id}'$ . The decryption of the ciphertext with the functional secret key succeeds if the identities match, in which case the inner-product  $\llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_t$  is recovered. Nothing else is revealed about the encrypted vector  $\mathbf{x}$ . However, we do not require that the identities  $\text{id}$  and  $\text{id}'$

or the vector  $\llbracket \mathbf{y} \rrbracket_2$  remain hidden. These functional secret keys can be generated from the master secret key of the FE scheme.

As we explained, the MA-ABE ciphertext will contain the FE encryption of the vector  $(s_j, u_j)$  for the identity  $\rho(j)$ , under the  $\text{FE.pk}$  of the authority that owns attribute  $\rho(j)$ , for all  $j \in [\ell]$ .

The secret key of a user identified by a global identifier  $\text{gid}$ , for an attribute  $\text{att}$ , will contain the FE functional secret key for the vector  $\llbracket (1, z_{\text{gid}}) \rrbracket_2$  and the identity  $\text{att}$ , where  $\llbracket z_{\text{gid}} \rrbracket_2$  is the output of the hash value  $H(\text{gid})$ . This FE functional secret key is computed using the FE master secret key of the authority that owns the attribute  $\text{att}$ .

The user  $\text{gid}$  collects all the FE functional secret keys  $\text{sk}_{\llbracket (1, z_{\text{gid}}) \rrbracket_2, \text{att}}$ , by making a request  $(\text{att}, \text{gid})$  to the relevant authorities. Each FE key  $\text{sk}_{\llbracket (1, z_{\text{gid}}) \rrbracket_2, \text{att}}$  yields the value  $\llbracket s_j + z_{\text{gid}} u_j \rrbracket_{\text{t}}$  if  $j$  is such that  $\rho(j) = \text{att}$ . If sufficiently many such values are revealed, then they can be combined to obtain  $\llbracket s + z_{\text{gid}} \cdot 0 \rrbracket_{\text{t}} = \llbracket s \rrbracket_{\text{t}}$ , the encapsulation key. Otherwise said, if the user  $\text{gid}$  possesses enough attributes to satisfy the MSP in the ciphertext, it recovers the encapsulation key. Here we rely on the fact that the share reconstruction for an MSP is linear.

To argue security, we could simply rely on the simulation security of the underlying FE scheme, which states that only the value  $\llbracket s_j + z_{\text{gid}} u_j \rrbracket_{\text{t}}$  is revealed by the ciphertext and the FE functional secret key for identity  $\rho(j)$  and vector  $\llbracket (1, z_{\text{gid}}) \rrbracket_2$  (together with the value  $\llbracket z_{\text{gid}} \rrbracket_2$ , which is public). Note that the term  $\llbracket z_{\text{gid}} u_j \rrbracket_{\text{t}}$  prevent collusions across different  $\text{gid}$ . In fact it hides the share  $s_j$ , assuming the values  $\llbracket z_{\text{gid}} \rrbracket_2$  generated by the hash function are pseudo-random (this holds in the random oracle model). So, if for any given  $\text{gid}$  there are not enough attributes to satisfy the access structure associated to the ciphertext, then there are not enough values  $\llbracket s_j + z_{\text{gid}} u_j \rrbracket_{\text{t}}$  to recover  $\llbracket s \rrbracket_{\text{t}}$ , which remains hidden.

This approach works, but it requires an FE scheme that is simulation-secure with many challenge ciphertexts. Unfortunately, such primitive cannot be built from standard assumptions (this can be proved by an incompressibility argument, similar to [BSW11], see Remark 1). We use an FE with indistinguishability-based security instead, which means that our MA-ABE requires a more sophisticated security proof relying on some prime-order variant of the dual vector pairing space methodology [OT09, Lew12]. Our modular construction can be instantiated with any FE with indistinguishability-based security for the appropriate functionality, such as the scheme from [ACGU20].

We now explain how to handle non-monotonic access structures, represented by span programs where each share is associated with either a normal or negated attribute (as per Definition 2). For negated attributes, we simply replace the identity-based FE for inner-products (which we call  $\mathcal{FE}_1$  here) in our modular construction with an FE with revocations (called  $\mathcal{FE}_2$ ). That is, the ciphertext of  $\mathcal{FE}_2$  encrypts a vector  $\mathbf{x}$  together with an identity  $\text{id}$ , as before, but now each functional secret key is associated with a vector  $\llbracket \mathbf{y} \rrbracket_2$  and a set of identities  $\mathcal{S}$ . If  $\text{id} \notin \mathcal{S}$ , then the decryption recovers  $\llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_{\text{t}}$ . Else, no information is revealed about  $\mathbf{x}$  (although the identity  $\text{id}$ , the vector  $\llbracket \mathbf{y} \rrbracket_2$  and the set  $\mathcal{S}$  are not hidden).

We present a new construction for such an FE scheme whose selective security is proven under standard pairing assumptions (SXDH). Our modular MA-ABE for non-monotonic access structures uses  $\mathcal{FE}_1$  and  $\mathcal{FE}_2$  as follows. The encryption creates shares  $\{s_j\}_{j \in [\ell]}$  of a random value  $s$  and shares  $\{u_j\}_{j \in [\ell]}$  of 0 according to the span program that represents the access structure, as before. The novelty here is that each share  $j \in [\ell]$  is mapped to  $\rho(j)$  which is either a normal attribute, in which case the encryption encrypts the vector  $(s_j, u_j)$  with the identity  $\rho(j)$  using  $\mathcal{FE}_1$ ; or it is mapped to  $\rho(j)$  which is a negated attribute, in which case the encryption encrypts  $(s_j, u_j)$  with the identity  $\rho(j)$  but this time using  $\mathcal{FE}_2$ . Let  $\text{gid}$  be the global identifier of a user that possesses different sets of attributes  $\mathcal{S}_{\text{aut}} = \{\text{att}_1^{\text{aut}}, \dots, \text{att}_{n_{\text{aut}}}^{\text{aut}}\}$  each owned by a different authority  $\text{aut}$ . For each authority  $\text{aut}$ , the user collects the  $\mathcal{FE}_2$  functional secret key for the vector  $\llbracket(1, z_{\text{gid}})\rrbracket_2$  and the set  $\mathcal{S}_{\text{aut}}$ , together with a set of  $n_{\text{aut}}$   $\mathcal{FE}_1$  functional secret keys for the vector  $\llbracket(1, z_{\text{gid}})\rrbracket_2$  and the identity  $\text{att}_i^{\text{aut}}$  for  $i = 1, \dots, n_{\text{aut}}$ . Thanks to these keys, a user can recover the values  $\llbracket s_j + z_{\text{gid}} u_j \rrbracket_t$  for the shares  $j$  associated with  $\rho(j)$  which is either a normal attribute owned by the user  $\text{gid}$ , or a negated attribute that is not part of the set of attributes owned by  $\text{gid}$ . As a result, decryption succeeds if and only if the attributes of the user  $\text{gid}$  satisfy the non-monotonic access structure. The security of the MA-ABE boils down to the security of the underlying FE schemes.

To build the FE for inner-products with revocations, we start with a one-time statistically secure scheme where the encryption of a vector  $\mathbf{x} \in \mathbb{Z}_p^n$  for an identity  $\text{id}^* \in \mathbb{Z}_p$  is of the form  $\text{ct} = (\mathbf{x} + \mathbf{v}, P(\text{id}^*))$  where  $\mathbf{v} \in \mathbb{Z}_p^n$  is a random vector and  $P$  is a random polynomial evaluated on  $\text{id}^* \in \mathbb{Z}_p$ . The functional secret key for a vector  $\mathbf{y} \in \mathbb{Z}_p^n$  and a set of identities  $\mathcal{S} \subset \mathbb{Z}_p$  is of the form  $\text{sk}_{\mathbf{y}, \mathcal{S}} = (\mathbf{y}^\top \mathbf{v} + P(0), \{P(\text{id})\}_{\text{id} \in \mathcal{S}})$ . We assume the identity space is  $\mathbb{Z}_p^*$ , excluding 0 as a valid identity. Polynomial  $P$  is of degree  $d$ , and we assume the set  $\mathcal{S}$  associated with each functional secret key is of size exactly  $d$ . We explain later how to remove this restriction. If  $\text{id}^* \notin \mathcal{S}$ , we have the evaluation of the polynomial  $P$  on  $d+1$  distinct points, so we can recover  $P(0)$  using Lagrange interpolation and get  $\mathbf{y}^\top \mathbf{v}$ , thanks to which we can obtain  $\mathbf{x}^\top \mathbf{y}$ . On the other hand, if  $\text{id}^* \in \mathcal{S}$ , we only have the evaluation of  $P$  on  $d$  distinct points, which reveals no information about  $P(0)$ , which completely masks  $\mathbf{v}^\top \mathbf{y}$ . Therefore,  $\mathbf{v}$  masks  $\mathbf{x}$  perfectly. To obtain an FE scheme with public-key encryption and security for many functional secret keys, we use standard techniques from pairing groups:

- instead of using the vector  $\mathbf{v}$  and the polynomial  $P$ , the encryption uses  $\llbracket \mathbf{v} \rrbracket_1$  and the coefficients of  $P$  in  $\mathbb{G}_1$  that are part of  $\text{pk}$  to compute:

$$\text{ct} = (\llbracket \mathbf{x} + \mathbf{v}r \rrbracket_1, \llbracket rP(\text{id}^*) \rrbracket_1) , \text{ for } r \leftarrow_R \mathbb{Z}_p .$$

- to obtain security against collusions, we randomize the functional secret keys:

$$\text{sk}_{\mathbf{y}, \mathcal{S}} = (\llbracket \mathbf{y}^\top \mathbf{v} + sP(0) \rrbracket_2, \{\llbracket sP(\text{id}) \rrbracket_2\}_{\text{id} \in \mathcal{S}}) , \text{ for } s \leftarrow_R \mathbb{Z}_p .$$

The scheme describe here would be secure in the generic group model. To accommodate for a security proof using the SXDH assumption (i.e. the assumption that

DDH holds both in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ ), we modify slightly the scheme using techniques reminiscent from the hash proof system from [CS02], similarly to [ALS16] in the context of functional encryption for inner-products.

*Remark 1 (Impossibility of simulation secure FE).* We consider an adversary playing against the many-ciphertexts simulation security of an identity-based FE scheme for inner-products, which makes  $q_1$  functional secret key queries for random vectors  $[\mathbf{y}_1]_2, \dots, [\mathbf{y}_{q_1}]_2$  and identities  $\text{id}_1, \dots, \text{id}_{q_1}$ . The adversary also chooses random vectors  $\mathbf{x}_1, \dots, \mathbf{x}_{q_2}$  and identities  $\text{id}_1^*, \dots, \text{id}_{q_2}^*$  for the challenge ciphertexts. The adversary chooses  $\text{id}_1 = \text{id}_2 = \dots = \text{id}_{q_1} = \text{id}_1^* = \dots = \text{id}_{q_2}^*$ . The simulator must produce the challenge ciphertexts and the functional secret keys using only the values  $[\mathbf{x}_i^\top \mathbf{y}_j]_t$  and  $[\mathbf{y}_j]_2$  for  $i = 1, \dots, q_2$  and  $j = 1, \dots, q_1$ , plus the identities. By the SXDH assumption (which we require for our MA-ABE), the  $q_1 \cdot q_2$  values  $[\mathbf{x}_i^\top \mathbf{y}_j]_t$  are pseudo-random. The ciphertexts and functional secret keys, which are of total size  $(q_1 + q_2) \cdot \text{poly}(\lambda)$  must encode these values of total size  $q_1 \cdot q_2 \cdot \text{poly}'(\lambda)$  where  $\text{poly}, \text{poly}'$  are polynomials, which is a contradiction. It is not clear how to bypass this impossibility result even in the random oracle model. In fact [AKW18] presents similar impossibility results for FE even in the random oracle model.

*Related Works.* [Kim19] builds a multi-authority ABE for all circuits from LWE for a slightly different notion that the GID model presented here (it can be seen as a relaxation of the GID model). In a recent work, [DKW21a] builds an MA-ABE for DNF formula from LWE, followed by [WWW22] that removed the use of random oracles. In [MJ18], the authors present a decentralized ABE, which is similar to an MA-ABE except the number of authorities of the system is fixed ahead of time, and each authority requires the public keys of the other authorities to generate its share of the user secret key. They realized this notion for the orthogonality-testing predicate (a.k.a. inner-product), which captures  $NC_0$  circuits. Later on, [Ayy22] extended their construction to partially hide the predicate in the user secret keys. In the same paper, they also presented a distributed ciphertext-policy ABE for  $NC_1$ , based on the LWE assumption and the bilinear generic group model. A distributed ABE is like an MA-ABE except the number of authorities is fixed ahead of time, and the adversary cannot create corrupted authorities with arbitrary public keys, but is instead restricted to (statically) recover the secret keys of honestly generated authorities. In [OT13], the authors build decentralized attribute-based signatures, which generalize the notion of ring signatures, by allowing a user whose attributes satisfy a predicate to sign a message with respect to the predicate. The validity of the signature implies that the signer has valid credentials, but the identity of the signer (or its attributes) remain hidden. As a side result, they also build a multi-authority ABE whose adaptive security is proven under the DLIN assumption in prime-order symmetric pairing groups in the random oracle model. Their scheme supports non-monotone access structures combined with inner-products. However, the security they prove does not handle corruptions of authorities. That is, in the security game, the adversary cannot get the secret key of a set of selected

authorities, as is the case for others multi-authority ABE. In a paper concurrent to our work [DKW21b], the authors give the first MA-ABE for monotone span programs from the *search* variant of the Bilinear Diffie Hellman assumption. In their scheme, the size of the MSP, the number of attribute re-use and the size of the attribute universe of each authorities are all a-priori bounded. Their construction also inherits some of the practical deficiencies from prior schemes, namely, it uses symmetric pairings and the ciphertexts contain many target group elements. In [WFL19], the authors build an MA-ABE for bounded collusions (that is, where the number of possible user secret keys that can be corrupted is a priori bounded). Their construction also relies on inner-product FE but which are not identity-based nor handle revocation. They can be built from DDH (without pairing). The main different with our work lies in the unbounded-collusion security feature we achieve, which requires different techniques.

## 2 Preliminaries

### 2.1 Notations

We say a function  $f : \mathbb{N} \rightarrow \mathbb{R}$  is negligible if  $f$  is asymptotically dominated by the inverse of any polynomial, i.e. for every polynomial  $p \in \mathbb{R}[X]$ , there exists  $\lambda_p \in \mathbb{N}$  such that  $|f(\lambda)| \leq |1/p(\lambda)|$  for all  $\lambda \geq \lambda_p$ . We denote by  $|\mathbf{v}|$  the length or dimension of vector  $\mathbf{v}$  and by  $v_i$  its  $i$ -th component. For any  $n \in \mathbb{N}$ , we denote  $\{1, \dots, n\}$  by  $[n]$ . For any column vector  $\mathbf{u} \in \mathbb{Z}^n$  and  $\mathbf{v} \in \mathbb{Z}^m$ , we denote by  $(\mathbf{v}, \mathbf{u}) \in \mathbb{Z}^{n+m}$  the column vector obtained by concatenating them. Given two matrices (or vectors)  $\mathbf{A} \in \mathbb{Z}^{m_1 \times n_1}$  and  $\mathbf{B} \in \mathbb{Z}^{m_2 \times n_2}$ , we denote by  $\mathbf{A} \otimes \mathbf{B} \in \mathbb{Z}^{m_1 m_2 \times n_1 n_2}$  their Kronecker product, aka. tensor product defined as follows. For all  $i \in [m_1 m_2]$  and  $j \in [n_1 n_2]$  which we can write  $i = m_1 i_1 + i_2$  with  $i_1 \in [m_2]$ ,  $i_2 \in [m_1]$ ,  $j = n_1 j_1 + j_2$  with  $j_1 \in [n_2]$ ,  $j_2 \in [n_1]$ , the  $(i, j)$ 'th coordinate of  $\mathbf{A} \otimes \mathbf{B}$  is  $a_{i_1, j_1} \cdot b_{i_2, j_2}$ .

### 2.2 Lagrange Interpolation

Let  $p$  be a prime and  $\mathbb{Z}_p[X]$  denotes the mono-variate polynomials over  $\mathbb{Z}_p$ . There exists an efficient deterministic algorithm **Lagr** such that for all  $P \in \mathbb{Z}_p[X]$  of degree  $d$ , given as input  $d + 1$  distinct values  $x_1, \dots, x_{d+1} \in \mathbb{Z}_p \setminus \{0\}$ , outputs  $(\alpha_1, \dots, \alpha_{d+1}) = \mathbf{Lagr}(x_1, \dots, x_{d+1})$  such that  $\alpha_i \in \mathbb{Z}_p$  for all  $i \in [d + 1]$  and  $P(0) = \sum_{i=1}^{d+1} \alpha_i P(x_i)$ . The following fact states that when the evaluations of a polynomial  $P$  of degree  $d$  at only  $d$  or less distinct points (different from 0) are given, it is impossible to recover the value  $P(0)$ , because it is statistically independent from the values at the other points.

*Fact 1.* Let  $d \in \mathbb{N}$ ,  $p$  be a prime,  $x_1, \dots, x_d \in \mathbb{Z}_p \setminus \{0\}$  be  $d$  distinct values and  $P$  be a uniformly random polynomial over  $\mathbb{Z}_p[X]$  of degree  $d$ . The value  $P(0)$  is statistically independent from  $\{P(x_1), \dots, P(x_d)\}$ .

### 2.3 Access Structure

We recall the definition of monotonic access structures using the language of monotonic span programs [KW93], which capture Boolean formulas. The set of all possible attributes used by an access structure is referred to as the attribute universe. Most of the prior works consider attribute universes of polynomial size (aka small universe) or at least attribute universe of finite size (aka large universe). Here we focus on unbounded attribute universe, where any bit string can serve as an attribute. This is the most advantageous setting in term of flexibility. We denote the set of all possible bit strings by  $\{0, 1\}^*$ .

**Definition 1 (Monotonic access structure [Bei96, KW93]).** A monotonic access structure is a pair  $(\mathbf{M}, \rho)$  where  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$  and  $\rho : [\ell] \rightarrow \{0, 1\}^*$ . The matrix  $\mathbf{M}$  is used to generate shares as described in Fig. 1, and  $\rho$  maps each share to its associated attribute. Given a set of attributes  $\mathcal{S} \subseteq \{0, 1\}^*$ , we say that

$$\mathcal{S} \text{ satisfies } (\mathbf{M}, \rho) \text{ iff } \mathbf{1} \in \text{Span}(\mathbf{M}_{\mathcal{S}}),$$

where  $\mathbf{1} := (1, 0, \dots, 0) \in \mathbb{Z}^n$ ;  $\mathbf{M}_{\mathcal{S}}$  denotes the collection of vectors  $\{\mathbf{M}_j : \rho(j) \in \mathcal{S}\}$  where  $\mathbf{M}_j$  denotes the  $j$ 'th column of  $\mathbf{M}$ ; and  $\text{Span}$  refers to linear span of collection of vectors over  $\mathbb{Z}_p$ .

That is,  $\mathcal{S}$  satisfies  $(\mathbf{M}, \rho)$  iff there exists constants  $\omega_1, \dots, \omega_\ell \in \mathbb{Z}_p$  such that

$$\sum_{\rho(j) \in \mathcal{S}} \omega_j \mathbf{M}_j = \mathbf{1} \tag{1}$$

Observe that the constants  $\{\omega_i\}$  can be computed in time polynomial in the size of the matrix  $\mathbf{M}$  via Gaussian elimination.

<p>Share <math>(\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}, \mathbf{a} \in \mathbb{Z}_p^d)</math>:</p> <p>Sample <math>\mathbf{U} \leftarrow_R \mathbb{Z}_p^{d \times (n-1)}</math>, and for all <math>j \in [\ell]</math>, set <math>\mathbf{a}_j := (\mathbf{a}   \mathbf{U}) \mathbf{M}_j \in \mathbb{Z}_p^d</math>.</p> <p>Return <math>\{\mathbf{a}_j\}_{j \in [\ell]}</math>.</p>
--

**Fig. 1.** Share generation algorithm. Here,  $\mathbf{M}_j$  denotes the  $j$ -th column of  $\mathbf{M}$ . For each  $j \in [\ell]$ ,  $\mathbf{a}_j$  is a share of the secret  $\mathbf{a} \in \mathbb{Z}_p^d$ .

Now we consider non-monotonic access structures, where  $\rho$  maps each share to either an attribute or a *negated* attribute. A set of attribute  $\mathcal{S}$  satisfies the non-monotonic access structure  $(\mathbf{M}, \rho)$  if given all the shares that correspond to an attribute in  $\mathcal{S}$  or a negated attribute of the form  $\neg \text{att}$  where  $\text{att}$  is not in  $\mathcal{S}$ , it is possible to recover the secret. For any set  $\mathcal{S} \subseteq \{0, 1\}^*$ , we denote by  $\{\neg\} \cdot \mathcal{S}$  the set defined as  $\{\neg \text{att}\}_{\text{att} \in \mathcal{S}}$ . The formal definition of a non-monotonic access structure is given below.

**Definition 2 (Non-monotonic access structure [OSW07]).** A non-monotonic access structure is a pair  $(\mathbf{M}, \rho)$  where  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$  and  $\rho : [\ell] \rightarrow \{0, 1\}^* \cup (\{\neg\} \cdot \{0, 1\}^*)$ . The matrix  $\mathbf{M}$  is used to generate shares as described in Fig. 1,



and  $\rho$  maps each share to its associated attribute in  $\{0, 1\}^*$  or negated attribute in  $\{\neg\} \cdot \{0, 1\}^*$ . Given a set of attributes  $\mathcal{S} \subseteq \{0, 1\}^*$ , we say that

$\mathcal{S}$  satisfies  $(\mathbf{M}, \rho)$  iff  $\mathbf{1} \in \text{Span}(\mathbf{M}_{\mathcal{S}})$ ,

where  $\mathbf{1} := (1, 0, \dots, 0) \in \mathbb{Z}^n$ ;  $\mathbf{M}_{\mathcal{S}}$  denotes the collection of vectors  $\{\mathbf{M}_j : \rho(j) \in \mathcal{S} \text{ or } \rho(j) = \neg\text{att with att} \in \{0, 1\}^* \setminus \mathcal{S}\}$ ,  $\mathbf{M}_j$  denotes the  $j$ 'th column of  $\mathbf{M}$ , and  $\text{Span}$  refers to the linear span of a collection of (column) vectors over  $\mathbb{Z}_p$ .

For any set of attributes  $\mathcal{S}_{\text{corr}} \subset \{0, 1\}^*$ , we say

$\mathcal{S}$  satisfies  $(\mathbf{M}, \rho)$  with corruptions  $\mathcal{S}_{\text{corr}}$  iff  $\mathbf{1} \in \text{Span}(\mathbf{M}_{\mathcal{S}, \mathcal{S}_{\text{corr}}})$ ,

where  $\mathbf{1} := (1, 0, \dots, 0) \in \mathbb{Z}^n$ ;  $\mathbf{M}_{\mathcal{S}}$  denotes the collection of vectors  $\{\mathbf{M}_j : \rho(j) \in \mathcal{S} \cup \mathcal{S}_{\text{corr}} \text{ or } \rho(j) = \neg\text{att with att} \in \{0, 1\}^* \setminus \mathcal{S}\}$ ,  $\mathbf{M}_j$  denotes the  $j$ 'th column of  $\mathbf{M}$ , and  $\text{Span}$  refers to the linear span of a collection of (column) vectors over  $\mathbb{Z}_p$ .

That is,  $\mathcal{S}$  satisfies  $(\mathbf{M}, \rho)$  iff there exists constants  $\omega_1, \dots, \omega_\ell, \omega'_1, \dots, \omega'_\ell \in \mathbb{Z}_p$  such that

$$\sum_{\rho(j) \in \mathcal{S} \cup \mathcal{S}_{\text{corr}}} \omega_j \mathbf{M}_j + \sum_{\rho(j) = \neg\text{att}, \text{att} \notin \mathcal{S}} \omega'_j \mathbf{M}_j = \mathbf{1} \quad (2)$$

Observe that the constants  $\{\omega_i, \omega'_i\}$  can be computed in time polynomial in the size of the matrix  $\mathbf{M}$  via Gaussian elimination. Now we recall a useful fact about access structures represented by span programs.

**Lemma 1** ([KW93]). *Let  $(\mathbf{M}, \rho)$  be a non-monotonic access structure where  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$ . For all sets  $\mathcal{S}, \mathcal{S}_{\text{corr}} \subseteq \{0, 1\}^*$  such that  $\mathcal{S}$  does not satisfy  $(\mathbf{M}, \rho)$  with corruptions  $\mathcal{S}_{\text{corr}}$ , there exists a vector  $\mathbf{w}_{\mathcal{S}} \in \mathbb{Z}_p^{\ell-1}$  such that  $(\mathbf{1}, \mathbf{w})^\top \mathbf{M}_j = 0$  for all  $j \in [\ell]$  such that  $\rho(j) \in \mathcal{S} \cup \mathcal{S}_{\text{corr}}$  or  $\rho(j) = \neg\text{att with att} \in \{0, 1\}^* \setminus \mathcal{S}$ .*

## 2.4 Pairing Groups

Let  $\text{GGen}$  be a PPT algorithm that on input the security parameter  $1^\lambda$ , outputs a description  $\mathcal{PG} = (p, \mathbb{G}_1, \mathbb{G}_2, P_1, P_2, \mathbb{G}_t, e)$  of pairing groups where  $\mathbb{G}_1, \mathbb{G}_2$  and  $\mathbb{G}_t$  are cyclic groups of order  $p$  for a  $2\lambda$ -bit prime  $p$ ;  $P_1$  and  $P_2$  are generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively and  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$  is an efficiently computable (non-degenerate) bilinear map, thus  $P_t := e(P_1, P_2)$  generates  $\mathbb{G}_t$ .

We use implicit representation of group elements. For  $s \in \{1, 2, t\}$  and  $a \in \mathbb{Z}_p$ , define  $\llbracket a \rrbracket_s = a \cdot P_s \in \mathbb{G}_s$  as the implicit representation of  $a$  in  $\mathbb{G}_s$ . More generally, for a matrix  $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$  we define  $\llbracket \mathbf{A} \rrbracket_s$  as the implicit representation of  $\mathbf{A}$  in  $\mathbb{G}_s$ :

$$\llbracket \mathbf{A} \rrbracket_s := \begin{pmatrix} a_{11} \cdot P_s & \dots & a_{1m} \cdot P_s \\ \vdots & & \vdots \\ a_{n1} \cdot P_s & \dots & a_{nm} \cdot P_s \end{pmatrix} \in \mathbb{G}_s^{n \times m}.$$

Given  $\llbracket a \rrbracket_1$  and  $\llbracket b \rrbracket_2$ , one can efficiently compute  $\llbracket a \cdot b \rrbracket_t$  using the pairing  $e$ . For matrices  $\mathbf{A}$  and  $\mathbf{B}$  of matching dimensions, define  $e(\llbracket \mathbf{A} \rrbracket_1, \llbracket \mathbf{B} \rrbracket_2) := \llbracket \mathbf{AB} \rrbracket_t$ . For any matrix  $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_p^{n \times m}$ , any group  $s \in \{1, 2, t\}$ , we denote by  $\llbracket \mathbf{A} \rrbracket_s + \llbracket \mathbf{B} \rrbracket_s = \llbracket \mathbf{A} + \mathbf{B} \rrbracket_s$ .

**Definition 3 (DDH assumption).** For any adversary  $\mathcal{A}$ , any group  $s \in \{1, 2, t\}$  and any security parameter  $\lambda$ , let

$$\text{Adv}_{\mathbb{G}_s, \mathcal{A}}^{\text{DDH}}(\lambda) := |\Pr[1 \leftarrow \mathcal{A}(\mathcal{PG}, [\mathbf{a}]_s, [\mathbf{ar}]_s)] - \Pr[1 \leftarrow \mathcal{A}(\mathcal{PG}, [\mathbf{a}]_s, [\mathbf{u}]_s)]|,$$

where the probabilities are taken over  $\mathcal{PG} \leftarrow_R \text{GGen}(1^\lambda, d)$ ,  $\mathbf{a} \leftarrow_R \mathbb{Z}_p^2$ ,  $r \leftarrow_R \mathbb{Z}_p$ ,  $\mathbf{u} \leftarrow_R \mathbb{Z}_p^2$ , and the random coins of  $\mathcal{A}$ . We say DDH holds in  $\mathbb{G}_s$  if for all PPT adversaries  $\mathcal{A}$ ,  $\text{Adv}_{\mathbb{G}_s, \mathcal{A}}^{\text{DDH}}(\lambda)$  is a negligible function of  $\lambda$ .

**Definition 4 (SXDH assumption).** For any security parameter  $\lambda$  and any pairing group  $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow_R \text{GGen}(1^\lambda)$ , we say SXDH holds in  $\mathcal{PG}$  if DDH holds in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ .

It is well known that the DDH and SXDH assumptions are equivalent when the dimensions of the vectors are larger than 2 (for any polynomially large dimensions).

### 2.5 Functional Encryption

We recall the notion of functional encryption originally given in [BSW11]. Let  $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  be a family of sets, where for each  $\lambda \in \mathbb{N}$ ,  $\mathcal{F}_\lambda$  is a set of functions from the message space  $\mathcal{X}_\lambda$  to the output space  $\mathcal{Y}_\lambda$ . A functional encryption scheme for  $\mathcal{F}$  consists of the following PPT algorithms.

- $\text{Setup}(1^\lambda) \rightarrow (\text{msk}, \text{pk})$ . On input the global parameters  $\text{gp}$ , it outputs a master secret key  $\text{msk}$  and a public key  $\text{pk}$ . The public key is (sometimes implicitly) input to all other algorithms.
- $\text{Enc}(\text{pk}, m) \rightarrow \text{ct}$ . On input the public key  $\text{pk}$  and a message  $m \in \mathcal{X}_\lambda$ , it outputs a ciphertext  $\text{ct}$ .
- $\text{KeyGen}(\text{msk}, f) \rightarrow \text{sk}_f$ . On input the master secret key  $\text{msk}$  and a function  $f \in \mathcal{F}_\lambda$ , it outputs a functional secret key  $\text{sk}_f$ , which includes the description of the function  $f$ .
- $\text{Dec}(\text{pk}, \text{ct}, \text{sk}_f) \rightarrow m$ . On input the public key  $\text{pk}$ , a ciphertext  $\text{ct}$  and a functional secret key  $\text{sk}_f$ , the decryption algorithm deterministically outputs a value  $\mu \in \mathcal{Y}_\lambda$  (or a special rejection symbol if it fails to decrypt).

**Correctness.** For all  $\lambda \in \mathbb{N}$ , all  $(\text{pk}, \text{msk})$  in the support of  $\text{Setup}(1^\lambda)$ , all messages  $m \in \mathcal{X}_\lambda$  and all functions  $f \in \mathcal{F}_\lambda$ , we have

$$\Pr[\text{Dec}(\text{pk}, \text{Enc}(\text{pk}, m), \text{KeyGen}(\text{msk}, f)) = f(m)] = 1,$$

where the probability is taken over the random coins of  $\text{Enc}$  and  $\text{KeyGen}$ .

We now describe the indistinguishability-based security notion for FE.

**Adaptive Security.** Given an FE scheme denoted by FE for  $\mathcal{F}$ , for any adversary  $\mathcal{A}$  and security parameter  $\lambda$ , we define the advantage function:

$$\text{Adv}_{\mathcal{A}}^{\text{FE}}(\lambda) := \left| \Pr \left[ \begin{array}{l} (\text{pk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda) \\ (m_0, m_1, \text{st}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{KeyGen}}(\cdot)}(\text{pk}) \\ \beta \leftarrow_R \{0, 1\} \\ \text{ct}^* \leftarrow \text{Enc}(\text{pk}, m_\beta) \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{KeyGen}}(\cdot)}(\text{ct}^*, \text{st}) \end{array} : \beta' = \beta \right] - \frac{1}{2} \right| ,$$

where the oracle  $\mathcal{O}_{\text{KeyGen}}$ , when given as input a function  $f \in \mathcal{F}_\lambda$ , returns  $\text{KeyGen}(\text{msk}, f)$  and  $\text{st}$  denotes the state of the adversary  $\mathcal{A}$ . We say the adversary  $\mathcal{A}$  is admissible if for all functions  $f \in \mathcal{F}_\lambda$  queried to  $\mathcal{O}_{\text{KeyGen}}$ , it holds that  $f(m_0) = f(m_1)$ . An FE scheme FE is said to be IND-secure if for all PPT admissible adversaries  $\mathcal{A}$ ,  $\text{Adv}_{\mathcal{A}}^{\text{FE}}$  is negligible.

**Selective, Super-Selective Security.** In the security game above, we say an adversary is selective if it chooses a pair of messages  $(m_0, m_1)$  before querying any functional secret key to  $\mathcal{O}_{\text{KeyGen}}$ . An adversary is said to be super-selective if it is selective and it chooses the queries to  $\mathcal{O}_{\text{KeyGen}}$  independently of the challenge ciphertext  $\text{ct}^*$ . That is, an FE scheme FE is said to be super-selective if for all admissible PPT adversaries  $\mathcal{A}$ , the function  $\text{Adv}_{\mathcal{A}}^{\text{ssel-FE}}$  is negligible, where  $\text{Adv}_{\mathcal{A}}^{\text{ssel-FE}}$  is defined for all  $\lambda \in \mathbb{N}$  as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{ssel-FE}}(\lambda) := \left| \Pr \left[ \begin{array}{l} (\text{pk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda) \\ (m_0, m_1, \text{st}) \leftarrow \mathcal{A}(\text{pk}) \\ \text{st}' \leftarrow \mathcal{A}(\text{st})^{\mathcal{O}_{\text{KeyGen}}(\cdot)} \\ \beta \leftarrow_R \{0, 1\} \\ \text{ct}^* \leftarrow \text{Enc}(\text{pk}, m_\beta) \\ \beta' \leftarrow \mathcal{A}(\text{ct}^*, \text{st}') \end{array} : \beta' = \beta \right] - \frac{1}{2} \right| ,$$

where the oracle  $\mathcal{O}_{\text{KeyGen}}$ , when given as input a function  $f \in \mathcal{F}_\lambda$ , returns  $\text{KeyGen}(\text{msk}, f)$  and  $\text{st}, \text{st}'$ s denote the states of the adversary  $\mathcal{A}$ . As for the IND-security above, we say the adversary  $\mathcal{A}$  is admissible if for all functions  $f \in \mathcal{F}_\lambda$  queried to  $\mathcal{O}_{\text{KeyGen}}$ , it holds that  $f(m_0) = f(m_1)$ .

## 2.6 Definition of Multi-authority ABE

We recall the definition of multi-authority ABE from [LW11]. We assume every authority is identified by a public key. For every authority  $\text{pk}$ , we denote by  $\mathcal{U}_{\text{pk}}$  the associated attribute universe. Without loss of generality, we assume that attribute universes are disjoint for different authorities.

We consider access structures  $(\mathbf{M}, \rho)$  where  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$ , and  $\rho$  maps each row  $j \in [\ell]$  to an attribute in  $\mathcal{U}_{\theta(j)}$ , where  $\theta$  maps a row  $j \in [\ell]$  to the authority who owns the attribute  $\rho(j)$ . To keep notations simple, we assume the map  $\theta$  is implicitly part of the description of the access structure.

**Definition.** A MA-ABE scheme consists of the following PPT algorithms:

- **GlobalSetup** $(1^\lambda) \rightarrow \text{gp}$ . On input the security parameter, it outputs global parameters, which are input to all other algorithms (usually implicitly).
- **AuthSetup** $(\text{gp}) \rightarrow (\text{pk}, \text{sk})$ . Each authority runs a setup procedure to generate its own pair of keys. The public key serves as a univocal identifier for the authority, which is associated with an attribute universe denoted by  $\mathcal{U}_{\text{pk}}$ .
- **Enc** $(\mathbf{M}, \rho, \Pi) \rightarrow (\text{ct}, \kappa)$ . On input an access structure  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$ ,  $\rho : [\ell] \rightarrow \{0, 1\}^*$  and a set of authorities  $\Pi$  such that for all columns  $j \in [\ell]$ , we have  $\theta(j) \in \Pi$ , the encryption algorithm outputs a ciphertext  $\text{ct}$  and a symmetric encryption key  $\kappa \in \mathcal{K}$ . The ciphertext implicitly contains a description of the access structure  $(\mathbf{M}, \rho)$ .
- **KeyGen** $(\text{pk}, \text{sk}, \text{gid}, \mathcal{S}) \rightarrow \text{sk}_{\text{gid}, \mathcal{S}}$ . On input an authority's public key  $\text{pk}$  and the corresponding secret key  $\text{sk}$ , a global identifier  $\text{gid}$  and a set of attribute  $\mathcal{S} \subset \mathcal{U}_{\text{pk}}$ , the key generation algorithm outputs a user secret key  $\text{sk}_{\text{gid}, \mathcal{S}}$ , which implicitly contains a description of  $\text{gid}$  and  $\mathcal{S}$ .
- **Dec** $(\text{ct}, \{\text{sk}_{\text{gid}, \mathcal{S}_i}\}_i) \rightarrow \kappa / \perp$ . On input a ciphertext  $\text{ct}$  and a set of user secret keys  $\{\text{sk}_{\text{gid}, \mathcal{S}_i}\}_i$  created for the same global identifier, the decryption algorithm deterministically outputs a symmetric key  $\kappa$  or  $\perp$ .

**Correctness.** For all  $\lambda \in \mathbb{N}$ , all  $\text{gp}$  in the support of **GlobalSetup** $(1^\lambda)$ , all  $\nu \in \mathbb{N}$ , all  $(\text{pk}_1, \text{sk}_1), \dots, (\text{pk}_\nu, \text{sk}_\nu)$  in the support of **Setup** $(\text{gp})$ , all access structures  $(\mathbf{M}, \rho)$  associated with the set of authorities  $\Pi = \{\text{pk}_1, \dots, \text{pk}_\nu\}$ , all pairs  $(\text{ct}, \kappa)$  in the support of **Enc** $(\mathbf{M}, \rho, \Pi)$ , all sets of attributes  $\mathcal{S}_i \subset \mathcal{U}_{\text{pk}_i}$  for all  $i \in [\nu]$  such that  $\mathcal{S} = \cup_{i \in [\nu]} \mathcal{S}_i$  satisfies  $(\mathbf{M}, \rho)$  and all global identifiers  $\text{gid} \in \{0, 1\}^*$ :

$$\Pr [\text{Dec}(\text{ct}, \{\text{sk}_{\text{gid}, \mathcal{S}_i}\}_{i \in [\nu]}) = \kappa] = 1 \text{ ,}$$

where the probability is taken over  $\text{sk}_{\text{gid}, \mathcal{S}_i} \leftarrow \text{KeyGen}(\text{pk}_i, \text{sk}_i, \text{gid}, \mathcal{S}_i)$  for all  $i \in [\nu]$ .

**Adaptive Security.** Given a multi-authority ABE denoted by ABE, for any stateful adversary  $\mathcal{A}$  and security parameter  $\lambda$ , we define the advantage function:

$$\text{Adv}_{\mathcal{A}}^{\text{ABE}}(\lambda) := \left| \Pr \left[ \begin{array}{l} \text{gp} \leftarrow \text{GlobalSetup}(1^\lambda) \\ (\mathbf{M}, \rho, \Pi_{\text{hon}}, \Pi_{\text{corr}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{create}}, \mathcal{O}_{\text{corr}}(\cdot), \mathcal{O}_{\text{KeyGen}}(\cdot, \cdot, \cdot)}(\text{gp}) \\ (\text{ct}^*, \kappa) \leftarrow \text{Enc}(\mathbf{M}, \rho, \Pi) \\ \beta \leftarrow_R \{0, 1\}; K_0 := \kappa; K_1 \leftarrow_R \mathcal{K} \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{corr}}(\cdot), \mathcal{O}_{\text{KeyGen}}(\cdot, \cdot, \cdot)}(\text{ct}^*, K_\beta) \end{array} : \beta' = \beta \right] - \frac{1}{2} \right|.$$

The oracles are defined as follows:

- $\mathcal{O}_{\text{create}}$ : runs  $(\text{pk}, \text{sk}) \leftarrow \text{AuthSetup}(\text{gp})$ , adds  $\text{pk}$  to the sets of honest authorities denoted by  $\mathcal{S}_{\text{hon}}$  (initially empty) and returns  $\text{pk}$ .
- $\mathcal{O}_{\text{corr}}(\text{pk})$ : if  $\text{pk} \in \mathcal{S}_{\text{hon}}$ , it returns the associated secret key  $\text{sk}$  and removes  $\text{pk}$  from  $\mathcal{S}_{\text{hon}}$ .
- $\mathcal{O}_{\text{KeyGen}}(\text{pk}, \text{gid}, \mathcal{S})$ : if  $\text{pk} \in \mathcal{S}_{\text{hon}}$  and  $\mathcal{S} \subset \mathcal{U}_{\text{pk}}$ , it returns  $\text{KeyGen}(\text{pk}, \text{sk}, \text{gid}, \mathcal{S})$  where  $\text{sk}$  is the secret key associated with  $\text{pk}$ ; otherwise, it returns  $\perp$ . This oracle can be queried at most once per  $(\text{pk}, \text{gid})$  pair. That is, there cannot be two queries of the form  $(\text{pk}, \text{gid}, \mathcal{S})$  and  $(\text{pk}, \text{gid}, \mathcal{S}')$  for different  $\mathcal{S} \neq \mathcal{S}'$  to  $\mathcal{O}_{\text{KeyGen}}$ . This restriction is necessary for non-monotonic access structure (see Remark 2).

The adversary  $\mathcal{A}$  outputs an access structure  $(\mathbf{M}, \rho)$  with respect to the authorities  $\Pi = \Pi_{\text{hon}} \cup \Pi_{\text{corr}}$ , where  $\Pi_{\text{hon}}$  denotes the set of honest authorities, that is, which have been created via  $\mathcal{O}_{\text{create}}$ , and which have not been queried to  $\mathcal{O}_{\text{corr}}$  (they can still be queried to  $\mathcal{O}_{\text{corr}}$  later on), whereas  $\Pi_{\text{corr}}$  denotes the set of corrupted authorities, that is, authorities created via  $\mathcal{O}_{\text{create}}$  that have been subsequently queried to  $\mathcal{O}_{\text{corr}}$ , or authorities whose public keys were maliciously created by the adversary  $\mathcal{A}$  himself. We require that  $\Pi_{\text{corr}}$  contains not only the public keys of the corrupted authorities, but also their associated secret keys<sup>1</sup>.

We denote by  $\mathcal{Q}_{\text{KeyGen}}$  the set of queries to  $\mathcal{O}_{\text{KeyGen}}$ ,  $\mathcal{S}_{\text{hon}} \subseteq \Pi_{\text{hon}}$  the set of authorities in  $\Pi_{\text{hon}}$  that are still honest at the end of the experiment,  $\mathcal{S}_{\text{corr}} = \Pi_{\text{corr}} \cup \Pi_{\text{hon}} \setminus \mathcal{S}_{\text{hon}}$ ,  $\Sigma_{\text{corr}} = \cup_{\text{pk} \in \mathcal{S}_{\text{corr}}} \mathcal{U}_{\text{pk}}$ , and for every global identifier  $\text{gid} \in \{0, 1\}^*$ ,  $\mathcal{S}_{\text{gid}} = \cup_{\text{pk} \in \mathcal{S}_{\text{hon}}, (\text{pk}, \text{gid}, \mathcal{S}) \in \mathcal{Q}_{\text{KeyGen}}} \mathcal{S}$ . We say the adversary  $\mathcal{A}$  is admissible if for all  $\text{gid} \in \{0, 1\}^*$ ,  $\mathcal{S}_{\text{gid}}$  does not satisfy  $(\mathbf{M}, \rho)$  with corruptions  $\Sigma_{\text{corr}}$  (as per Definition 1). We say ABE is adaptively secure if for all PPT admissible adversaries  $\mathcal{A}$ , there exists a negligible function  $\nu$  such that for all  $\lambda \in \mathbb{N}$ ,  $\text{Adv}_{\mathcal{A}}^{\text{ABE}}(\lambda) \leq \nu(\lambda)$ .

**Static Corruptions.** We say an ABE is secure with static corruptions if the adversary does not have access to the oracle  $\mathcal{O}_{\text{corr}}$ . He can still create authorities maliciously as part of  $\Pi_{\text{corr}}$ , but all authorities created by  $\mathcal{O}_{\text{create}}$  remain honest throughout the experiment.

<sup>1</sup> The restriction which requires that the adversary provide the secret keys of the corrupted authorities in  $\Pi_{\text{corr}}$  can be lifted via a generic use of Zero-Knowledge Argument of Knowledge. See Remark 3 for further details.

**Selective, Super-Selective Security.** In the security game above, we say an adversary is selective if it chooses the tuple  $(M, \rho, \Pi_{\text{corr}}, \Pi_{\text{hon}})$  before querying any user secret key to  $\mathcal{O}_{\text{KeyGen}}$ . An adversary is said to be super-selective if it is selective and it chooses the queries to  $\mathcal{O}_{\text{KeyGen}}$  independently of the challenge ciphertext  $\text{ct}^*$ . That is, an MA-ABE scheme ABE is said to be super-selective if for all admissible PPT adversaries  $\mathcal{A}$ , the function  $\text{Adv}_{\mathcal{A}}^{\text{ssel-ABE}}$  is negligible, where  $\text{Adv}_{\mathcal{A}}^{\text{ssel-ABE}}$  is defined for all  $\lambda \in \mathbb{N}$  as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{ABE}}(\lambda) := \Pr \left[ \begin{array}{l} \text{gp} \leftarrow \text{GlobalSetup}(1^\lambda) \\ (M, \rho, \Pi_{\text{hon}}, \Pi_{\text{corr}}, \text{st}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{create}}, \mathcal{O}_{\text{corr}}(\cdot)}(\text{gp}) \\ \text{st}' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{corr}}(\cdot), \mathcal{O}_{\text{KeyGen}}(\cdot, \cdot, \cdot)}(\text{st}) \\ (\text{ct}^*, \kappa) \leftarrow \text{Enc}(M, \rho, \Pi) \\ \beta \leftarrow_R \{0, 1\}; K_0 := \kappa; K_1 \leftarrow_R \mathcal{K} \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{corr}}(\cdot)}(\text{ct}^*, K_\beta, \text{st}') \end{array} : \beta' = \beta \right] - \frac{1}{2}.$$

where the oracles are defined as above, and  $\text{st}, \text{st}'$  denote the states of the adversary  $\mathcal{A}$ .

*Remark 2 (At most one user secret key query per gid).* In the definitions above, we restrict the adversary to query the oracle  $\mathcal{O}_{\text{KeyGen}}$  at most once per  $(\text{pk}, \text{gid})$  pair. This restriction is necessary when considering non-monotone access structure. In fact, security relies on the fact that users only obtain user secret keys associated to the set of *all* attributes they possess. Giving the adversary access to at most one query to  $\mathcal{O}_{\text{KeyGen}}$  per  $(\text{pk}, \text{gid})$  is one way to ensure this is the case.

For instance, suppose a user Alice possesses the attributes  $\text{att}_1$  and  $\text{att}_2$  that are owned by an authority. Alice should not be able to obtain user secret keys associated to strict subsets of  $\{\text{att}_1, \text{att}_2\}$ . If for example she obtains a user secret key for  $\{\text{att}_1\}$ , she would be able to decrypt a ciphertext associated with an access structure excluding users possessing  $\text{att}_2$ .

*Remark 3 (Stronger security via ZK-AoK).* In the security definition above, we require the adversary to provide not only the public keys, but also the secret keys of all the authorities in  $\Pi_{\text{corr}}$ . It is possible to lift this restriction, and thereby strengthen the security definition, using standard techniques involving Zero-Knowledge Argument of Knowledge (ZK-AoK). Any authority must publish not only a public key, but also an argument of knowledge of the associated secret key. The zero-knowledge property ensures that nothing is revealed about the secret key, and the argument of knowledge property forces the issuer to know the associated secret key. This way, the adversary must know the secret key associated to any authority it creates maliciously, since it has to provide an argument of knowledge. Note that in our ABE constructions we use a ZK-AoK for a very simple language that admits an efficient sigma protocol, that can be made non-interactive with the Fiat-Shamir heuristic. Consequently, strengthening the security comes at a modest efficiency cost. In the rest of this paper, we focus on the weaker security definition, which is easier to prove.

### 3 Inner-Product FE

#### 3.1 Identity-Based Inner-product FE

We recall the definition of Identity-Based Inner-Product Functional Encryption (ID-IPFE) which is a particular case of Functional Encryption where the family  $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  is as follows. Let  $d$  be a polynomial and  $\text{GGen}$  a pairing group generator. For every  $\lambda \in \mathbb{N}$ , the set of functions  $\mathcal{F}_\lambda$  is associated with a pairing group  $(p, \mathbb{G}_1, \mathbb{G}_2, P_1, P_2, \mathbb{G}_t, e) = \text{GGen}(1^\lambda)$ , where  $p$  is a prime which denotes the order of the groups  $\mathbb{G}_1, \mathbb{G}_2$ , and  $\mathbb{G}_t$ . We assume the pairing group  $\mathcal{PG}$  is given as input of the setup algorithm. The message space  $\mathcal{X}_\lambda = \mathbb{Z}_p^{d(\lambda)} \times \mathbb{Z}_p$ . That is, every message is of the form  $(\mathbf{x}, \text{id})$ , where  $\mathbf{x} \in \mathbb{Z}_p^{d(\lambda)}$  is referred to as the message vector, and  $\text{id} \in \mathbb{Z}_p$  is referred to as the identity. The function space  $\mathcal{F}_\lambda = \mathbb{G}_2^{d(\lambda)} \times \mathbb{Z}_p$ . Every function is of the form  $(\llbracket \mathbf{y} \rrbracket_2, \text{id}')$  where  $\llbracket \mathbf{y} \rrbracket_2 \in \mathbb{G}_2^{d(\lambda)}$  and  $\text{id}' \in \mathbb{Z}_p$ . Decryption recovers the inner product  $\llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_t \in \mathbb{G}_t$  when  $\text{id} = \text{id}'$ . When  $\text{id}' \neq \text{id}$ , the vector  $\mathbf{x}$  remains hidden. In both cases, the vector  $\llbracket \mathbf{y} \rrbracket_2$  and the identities  $\text{id}$  and  $\text{id}'$  are revealed.

In [DP19, TT18], the authors give an unbounded variant of the related family where functions are of the form  $(\mathbf{y}, \text{id}) \in \mathbb{Z}_p^{d(\lambda)} \times \mathbb{Z}_p$ , that is, the vector  $\mathbf{y}$  needs to be known in  $\mathbb{Z}_p^{d(\lambda)}$  instead of  $\mathbb{G}_2^{d(\lambda)}$ . In our MA-ABE that uses the ID-IPFE as a building block, the party generating the functional secret keys only know the value  $\llbracket \mathbf{y} \rrbracket_2 \in \mathbb{G}_2^{d(\lambda)}$ , which prevents us from using their scheme. In [ACGU20], the authors present an ID-IPFE for the functions described above (where  $\mathcal{F}_\lambda = \mathbb{G}_2^{d(\lambda)} \times \mathbb{Z}_p$ ) which is selectively secure under the SXDH assumption. They also present an adaptively secure construction but only for the messages  $(\mathbf{x}, \text{id})$  and functions  $(\llbracket \mathbf{y} \rrbracket_2, \text{id}')$  such that  $\mathbf{x}^\top \mathbf{y}$  is small (i.e. lies in a set of polynomial size), which is not the case for our application. Indeed the value of the inner product  $\llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_t$  in our case will be well-spread in the full group  $\mathbb{G}_t$ . This prevents from using the adaptively secure scheme from [ACGU20]. It is an open problem to build an adaptively secure ID-IPFE for large values.

#### 3.2 Inner-Product FE with Revocations

Here we consider a Functional Encryption scheme for the family  $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  where for all  $\lambda \in \mathbb{N}$ ,  $\mathcal{X}_\lambda = \mathbb{Z}_p^{d(\lambda)} \times \mathbb{Z}_p$ ,  $\mathcal{F}_\lambda = \mathbb{G}_2^{d(\lambda)} \times \mathcal{S}_t$ ,  $\mathcal{S}_t$  denotes all the sets of size  $t$  included in  $\mathbb{Z}_p$ , and  $p$  is a prime which denotes the order of a pairing group  $\mathcal{PG} = (p, \mathbb{G}_1, \mathbb{G}_2, P_1, P_2, \mathbb{G}_t, e)$ . We assume the pairing group  $\mathcal{PG}$  is given as input of the setup algorithm. For every message of the form  $(\mathbf{x}, \text{id})$  where  $\mathbf{x} \in \mathbb{Z}_p^{d(\lambda)}$  and  $\text{id} \in \mathbb{Z}_p$ , and every function of the form  $(\llbracket \mathbf{y} \rrbracket_2, \mathcal{S})$  where  $\mathcal{S} \subset \mathbb{Z}_p$  is of size  $t$ , decryption recovers  $\llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_t$  when  $\text{id} \notin \mathcal{S}$ . When  $\text{id} \in \mathcal{S}$ , then the vector  $\mathbf{x}$  remains hidden. In both cases, the identity  $\text{id}$ , the set  $\mathcal{S}$  and the vector  $\llbracket \mathbf{y} \rrbracket_2$  are revealed. Note that the set  $\mathcal{S}$  associated to each functional secret key is required to be of size *exactly*  $t$ . We argue in Sect. 3.3 how to remove this restriction and have sets of size at most  $t$ . We now give the first construction of such an FE

scheme, whose selective security we prove under SXDH. It is described in Fig. 2. It makes use of Lagrange interpolation, described in Sect. 2.2.

<p><b>Setup</b>(<math>1^\lambda, \mathcal{PG}</math>) :</p> <p>Given as input the security parameter <math>\lambda \in \mathbb{N}</math> and a pairing group <math>\mathcal{PG} := (p, \mathbb{G}_1, \mathbb{G}_2, P_1, P_2, \mathbb{G}_t, e)</math>, it samples <math>\mathbf{U}_0, \dots, \mathbf{U}_t \leftarrow_R \mathbb{Z}_p^{2 \times 2}</math>, <math>\mathbf{a}, \mathbf{b} \leftarrow_R \mathbb{Z}_p^2</math>, <math>\mathbf{V} \leftarrow_R \mathbb{Z}_p^{d \times 2}</math>.</p> <p>For all <math>x \in \mathbb{Z}_p</math>, we define <math>\mathbf{P}(x) = \mathbf{U}_0 + \mathbf{U}_1 x + \dots + \mathbf{U}_t x^t \in \mathbb{Z}_p^{2 \times 2}</math>.</p> <p>Set <math>\text{msk} = (\mathbf{V}, \llbracket \mathbf{b} \rrbracket_2, (\llbracket \mathbf{U}_i^\top \mathbf{b} \rrbracket_2)_{i \in \{0, \dots, t\}})</math>, <math>\text{pk} = (\llbracket \mathbf{a} \rrbracket_1, (\llbracket \mathbf{U}_i \mathbf{a} \rrbracket_1)_{i \in \{0, \dots, t\}}, \llbracket \mathbf{V} \mathbf{a} \rrbracket_1)</math> and output <math>(\text{msk}, \text{pk})</math>.</p>
<p><b>Enc</b>(<math>\text{pk}, \mathbf{x}, \text{id}</math>) :</p> <p>Given <math>\text{pk}</math>, <math>\mathbf{x} \in \mathbb{Z}_p^d</math>, <math>\text{id} \in \mathbb{Z}_p</math>, it samples <math>r \leftarrow_R \mathbb{Z}_p</math> and returns <math>\text{ct} = (\llbracket \mathbf{a} r \rrbracket_1, \llbracket \mathbf{x} + \mathbf{V} \mathbf{a} r \rrbracket_1, \llbracket \mathbf{P}(\text{id}) \mathbf{a} r \rrbracket_1) \in \mathbb{G}_1^{2+d+2}</math>.</p>
<p><b>KeyGen</b>(<math>\text{msk}, \llbracket \mathbf{y} \rrbracket_2, \mathcal{S}</math>):</p> <p>Given <math>\text{msk}</math>, <math>\llbracket \mathbf{y} \rrbracket_2 \in \mathbb{G}_2^d</math> and a set <math>\mathcal{S} \subset \mathbb{Z}_p</math> of size <math>t</math>, it samples <math>s \leftarrow_R \mathbb{Z}_p</math>, and returns <math>\text{sk} = (\llbracket \mathbf{b} s \rrbracket_2, \llbracket \mathbf{V}^\top \mathbf{y} + \mathbf{P}(0)^\top \mathbf{b} s \rrbracket_2, (\llbracket \mathbf{P}(\text{id}_j)^\top \mathbf{b} s \rrbracket_2)_{\text{id}_j \in \mathcal{S}}) \in \mathbb{G}_2^{2+2+2t}</math>.</p>
<p><b>Dec</b>(<math>\text{ct}, \text{id}, \text{sk}, \llbracket \mathbf{y} \rrbracket_2, \mathcal{S}</math>):</p> <p>Parse <math>\text{ct}</math> as <math>(\llbracket \mathbf{c}_1 \rrbracket_1, \llbracket \mathbf{c}_2 \rrbracket_1, \llbracket \mathbf{c}_3 \rrbracket_1) \in \mathbb{G}_1^2 \times \mathbb{G}_1^d \times \mathbb{G}_1^2</math>, <math>\text{sk}</math> as <math>(\llbracket \mathbf{k}_1 \rrbracket_2, \llbracket \mathbf{k}_2 \rrbracket_2, (\llbracket \mathbf{k}_{j,3} \rrbracket_2)_{j \in [t]}) \in \mathbb{G}_2^2 \times \mathbb{G}_2^2 \times (\mathbb{G}_1^t)^t</math>, and <math>\mathcal{S}</math> as <math>\mathcal{S} = \{\text{id}_1, \dots, \text{id}_t\}</math>.</p> <p>For all <math>j \in [t]</math>, compute <math>\llbracket \gamma_j \rrbracket_t = e(\llbracket \mathbf{c}_1^\top \rrbracket_1, \llbracket \mathbf{k}_{j,3} \rrbracket_2)</math> and <math>\llbracket \gamma_{t+1} \rrbracket_t = e(\llbracket \mathbf{c}_3^\top \rrbracket_1, \llbracket \mathbf{k}_1 \rrbracket_2)</math>.</p> <p>Compute <math>(\alpha_1, \dots, \alpha_{t+1}) = \text{Lagr}(\text{id}_1, \dots, \text{id}_t, \text{id})</math>. Return <math>e(\llbracket \mathbf{c}_2^\top \rrbracket_1, \llbracket \mathbf{y} \rrbracket_2) \cdot \prod_{j \in [t+1]} \llbracket \gamma_j \rrbracket_t^{\alpha_j} / e(\llbracket \mathbf{c}_1 \rrbracket_1, \llbracket \mathbf{k}_2 \rrbracket_2)</math>.</p>

**Fig. 2.** Inner-product FE with revocations for  $d$ -dimensional vectors and sets of size  $t$ . Its selective security is proven under SXDH. The algorithm **Lagr** is described in Sect. 2.2.

**Correctness.** Since  $\text{id} \notin \mathcal{S}$ , we can use the correctness of the algorithm **Lagr**, which states that:  $\prod_{j \in [t+1]} \llbracket \gamma_j \rrbracket_t^{\alpha_j} = \llbracket \mathbf{s} \mathbf{b}^\top \mathbf{P}(0) \mathbf{a} r \rrbracket_t$ . Thus, the decryption computes:

$$\begin{aligned}
 & e(\llbracket \mathbf{c}_2^\top \rrbracket_1, \llbracket \mathbf{y} \rrbracket_2) \cdot \prod_{j \in [t+1]} \llbracket \gamma_j \rrbracket_t^{\alpha_j} / e(\llbracket \mathbf{c}_1 \rrbracket_1, \llbracket \mathbf{k}_2 \rrbracket_2) \\
 &= \llbracket (\mathbf{x} + \mathbf{V} \mathbf{a} r)^\top \mathbf{y} + \mathbf{s} \mathbf{b}^\top \mathbf{P}(0) \mathbf{a} r - r \mathbf{a}^\top (\mathbf{V}^\top \mathbf{y} + \mathbf{P}(0)^\top \mathbf{b} s) \rrbracket_t \\
 &= \llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_t.
 \end{aligned}$$



**Theorem 1 (Selective security).** *The scheme presented in Fig. 2 is selectively secure under the SXDH assumption.*

*Proof.* We proceed via a series of hybrid games described bellow (the differences from one game to the next are highlighted in red).

Game<sub>0</sub> : is the game from the selective security definition in Sect. 2.5. Recall that the adversary  $\mathcal{A}$  first receives  $\text{pk} = \left( [\mathbf{a}]_1, (\llbracket \mathbf{U}_i \mathbf{a} \rrbracket_1)_{i \in \{0, \dots, t\}}, \llbracket \mathbf{V} \mathbf{a} \rrbracket_1 \right)$ . Then, it chooses a pair of messages  $((\mathbf{x}_0, \text{id}_0), (\mathbf{x}_1, \text{id}_1))$ , upon which it receives  $\text{ct}^* = (\llbracket \mathbf{a} \mathbf{r} \rrbracket_1, \llbracket \mathbf{x}_\beta + \mathbf{V} \mathbf{a} \mathbf{r} \rrbracket_1, \llbracket \mathbf{P}(\text{id}_\beta) \mathbf{a} \mathbf{r} \rrbracket_1)$ , where  $\beta \leftarrow_R \{0, 1\}$ . Afterwards, it can query its oracle  $\mathcal{O}_{\text{KeyGen}}$  on inputs of the form  $(\llbracket \mathbf{y} \rrbracket_2, \mathcal{S})$ , upon which it gets  $\text{sk} = (\llbracket \mathbf{b} \mathbf{s} \rrbracket_2, \llbracket \mathbf{V}^\top \mathbf{y} + \mathbf{P}(0)^\top \mathbf{b} \mathbf{s} \rrbracket_2, (\llbracket \mathbf{P}(\text{id}_j)^\top \mathbf{b} \mathbf{s} \rrbracket_2)_{\text{id}_j \in \mathcal{S}})$ . The adversary  $\mathcal{A}$  is admissible, which means that  $\text{id}_0 = \text{id}_1$ , which we denote by  $\text{id}^* = \text{id}_0 = \text{id}_1$ , and that for all queries  $(\llbracket \mathbf{y} \rrbracket_2, \mathcal{S})$  to  $\mathcal{O}_{\text{KeyGen}}$ , we have  $\text{id}^* \in \mathcal{S}$  or  $(\text{id}^* \notin \mathcal{S} \text{ and } \mathbf{x}_0^\top \mathbf{y} = \mathbf{x}_1^\top \mathbf{y})$ . At the end, the adversary  $\mathcal{A}$  outputs a guess  $\beta'$ .

Game<sub>1</sub> : we change the way the challenge ciphertext is computed. Namely, we have now

$$\text{ct}^* = (\llbracket \mathbf{z} \rrbracket_1, \llbracket \mathbf{x}_\beta + \mathbf{V} \mathbf{z} \rrbracket_1, \llbracket \mathbf{P}(\text{id}^*) \mathbf{z} \rrbracket_1) ,$$

where  $\mathbf{z} \leftarrow_R \mathbb{Z}_p^2$ . We prove that  $\text{Game}_0 \approx_c \text{Game}_1$  by the DDH assumption in  $\mathbb{G}_1$ . Namely, we have  $(\llbracket \mathbf{a} \rrbracket_1, \llbracket \mathbf{a} \mathbf{r} \rrbracket_1) \approx_c (\llbracket \mathbf{a} \rrbracket_1, \llbracket \mathbf{z} \rrbracket_1)$  where the leftmost distribution corresponds to  $\text{Game}_0$ , whereas the rightmost distribution corresponds to  $\text{Game}_1$ .

Game<sub>2</sub> : we change the way the challenge ciphertext is computed. Namely, we have now

$$\text{ct}^* = (\llbracket \mathbf{z} \rrbracket_1, \llbracket \mathbf{x}_\beta + \mathbf{V} \mathbf{z} \rrbracket_1, \llbracket \mathbf{P}(\text{id}^*) \mathbf{z} \rrbracket_1) ,$$

where  $\mathbf{z} \leftarrow_R \mathbb{Z}_p^2 \setminus \text{Span}(\mathbf{a})$ . Here  $\text{Span}(\mathbf{a})$  denotes the set of vectors proportional to  $\mathbf{a}$ . The cardinal of  $\text{Span}(\mathbf{a})$  is  $p$ , thus, the statistical distance between the uniform distribution over  $\mathbb{Z}_p^2 \setminus \text{Span}(\mathbf{a})$  and uniform over  $\mathbb{Z}_p^2$  is  $1/p$ , and  $\text{Game}_1 \approx_s \text{Game}_2$ .

Game<sub>3</sub> : we change the way the functional keys and the challenge ciphertext are computed. Namely, the ciphertext is now of the form:

$$\text{ct}^* = (\llbracket \mathbf{z} \rrbracket_1, \llbracket \mathbf{V} \mathbf{z} \rrbracket_1, \llbracket \mathbf{P}(\text{id}^*) \mathbf{z} \rrbracket_1) .$$

Note that the ciphertext does not depend on the message  $\mathbf{x}_\beta$  anymore. Each query  $(\llbracket \mathbf{y} \rrbracket_2, \mathcal{S})$  to  $\mathcal{O}_{\text{KeyGen}}$  is now answered with

$$\left( \llbracket \mathbf{b} \mathbf{s} \rrbracket_2, \llbracket \mathbf{V}^\top \mathbf{y} - \mathbf{a}^\perp \cdot \mathbf{x}_\beta^\top \mathbf{y} + \mathbf{P}(0)^\top \mathbf{b} \mathbf{s} \rrbracket_2, (\llbracket \mathbf{P}(\text{id}_j)^\top \mathbf{b} \mathbf{s} \rrbracket_2)_{\text{id}_j \in \mathcal{S}} \right) ,$$

where  $\mathbf{a}^\perp \in \mathbb{Z}_p^2$  is the vector such that  $\mathbf{a}^\top \mathbf{a}^\perp = 0$  and  $\mathbf{z}^\top \mathbf{a}^\perp = 1$ .  $\text{Game}_2$  and  $\text{Game}_3$  are identically distributed, since for all  $\mathbf{x}_\beta \in \mathbb{Z}_p^d$ , all  $\mathbf{a}^\perp \in \mathbb{Z}_p^2$ , the following are identically distributed:  $\{\mathbf{V} \leftarrow_R \mathbb{Z}_p^{d \times 2} : \mathbf{V}\}$  and  $\{\mathbf{V} \leftarrow_R \mathbb{Z}_p^{d \times 2} : \mathbf{V} - \mathbf{x}_\beta(\mathbf{a}^\perp)^\top\}$ . The former distribution corresponds to  $\text{Game}_2$  with some pre and post-processing, whereas the latter corresponds to  $\text{Game}_3$  with the same pre and post-processing. Note that  $\text{Game}_3$  crucially relies on the fact that the adversary is selective, since the vector  $\mathbf{x}_\beta$  needs to be known to generate all functional secret keys.

$\text{Game}_4$ : we change the way the functional keys are computed. Namely, each query  $(\llbracket \mathbf{y} \rrbracket_2, \mathcal{S})$  to  $\mathcal{O}_{\text{KeyGen}}$  is now answered with

$$\left( \llbracket \mathbf{b}_s \rrbracket_2, \llbracket \mathbf{V}^\top \mathbf{y} - \mathbf{1}_{\text{id}^* \notin \mathcal{S}} \mathbf{a}^\perp \mathbf{x}_\beta^\top \mathbf{y} + \mathbf{P}(0)^\top \mathbf{b}_s \rrbracket_2, (\llbracket \mathbf{P}(\text{id}_j)^\top \mathbf{b}_s \rrbracket_2)_{\text{id}_j \in \mathcal{S}} \right).$$

That is, now we only have the term  $\mathbf{a}^\perp \mathbf{x}_\beta^\top \mathbf{y}$  for functional key queries  $(\mathbf{y}, \mathcal{S})$  where  $\text{id}^* \notin \mathcal{S}$ . To transition from  $\text{Game}_3$  to  $\text{Game}_4$ , we use the following hybrid games.

$\text{Game}_{3,i}$ : for all  $i \in \{0, \dots, Q\}$ , where  $Q$  denotes the number of functional key queries,  $\text{Game}_{3,i}$  is defined as  $\text{Game}_4$  for the first  $i$ 'th key queries and as  $\text{Game}_3$  for the last  $Q - i$  queries. By definition we have  $\text{Game}_3 = \text{Game}_{3,0}$  and  $\text{Game}_4 = \text{Game}_{3,Q}$ . It suffices to show that for all  $i \in [Q]$ ,  $\text{Game}_{3,i-1} \approx_c \text{Game}_{3,i}$ . To do so, we introduce new intermediate games, defined as follows.

$\text{Game}_{3,i-1,1}$ : is defined as  $\text{Game}_{3,i-1}$ , except the  $i$ 'th query to  $\mathcal{O}_{\text{KeyGen}}$ , denoted by  $(\llbracket \mathbf{y}_i \rrbracket_2, \mathcal{S}_i)$ , is now answered with

$$\left( \llbracket \mathbf{d} \rrbracket_2, \llbracket \mathbf{V}^\top \mathbf{y}_i - \mathbf{a}^\perp \cdot \mathbf{x}_\beta^\top \mathbf{y}_i + \mathbf{P}(0)^\top \mathbf{d} \rrbracket_2, (\llbracket \mathbf{P}(\text{id}_j)^\top \mathbf{d} \rrbracket_2)_{\text{id}_j \in \mathcal{S}_i} \right),$$

where  $\mathbf{d} \leftarrow_R \mathbb{Z}_p^2$ . We have  $\text{Game}_{3,i-1} \approx_c \text{Game}_{3,i-1,1}$  by the DDH assumption in  $\mathbb{G}_2$ , which states that  $(\llbracket \mathbf{b} \rrbracket_2, \llbracket \mathbf{b} \mathbf{s}_i \rrbracket_2) \approx_c (\llbracket \mathbf{b} \rrbracket_2, \llbracket \mathbf{d} \rrbracket_2)$  where  $\mathbf{b}, \mathbf{d} \leftarrow_R \mathbb{Z}_p^2$ ,  $\mathbf{s}_i \leftarrow_R \mathbb{Z}_p$ . The former distribution corresponds to  $\text{Game}_{3,i-1}$  with some efficient post-processing, whereas the latter corresponds to  $\text{Game}_{3,i-1,1}$  with the same post-processing.

$\text{Game}_{3,i-1,2}$ : is defined as  $\text{Game}_{3,i-1,1}$ , except the vector  $\mathbf{d}$  used to compute the  $i$ 'th queried functional secret key is sampled as  $\mathbf{d} \leftarrow_R \mathbb{Z}_p^2 \setminus \text{Span}(\mathbf{b})$ , instead of uniformly random over  $\mathbb{Z}_p^2$ . Since the cardinal of  $\text{Span}(\mathbf{b})$  is at most  $p$ , the uniform distribution over  $\mathbb{Z}_p^2 \setminus \text{Span}(\mathbf{b})$  has statistical distance at most  $1/p$  with the uniform distribution over  $\mathbb{Z}_p^2$ . Thus,  $\text{Game}_{3,i-1,1} \approx_s \text{Game}_{3,i-1,2}$ .

$\text{Game}_{3,i-1,3}$ : is defined as  $\text{Game}_{3,i-1,2}$ , except the  $i$ 'th query to  $\mathcal{O}_{\text{KeyGen}}$  is now answered with

$$\left( \llbracket \mathbf{d} \rrbracket_2, \llbracket \mathbf{V}^\top \mathbf{y}_i - \mathbf{1}_{\text{id}^* \notin \mathcal{S}_i} \mathbf{a}^\perp \mathbf{x}_\beta^\top \mathbf{y}_i + \mathbf{P}(0)^\top \mathbf{d} \rrbracket_2, (\llbracket \mathbf{P}(\text{id}_j)^\top \mathbf{d} \rrbracket_2)_{\text{id}_j \in \mathcal{S}_i} \right),$$

where  $\mathbf{d} \leftarrow_R \mathbb{Z}_p^2 \setminus \text{Span}(\mathbf{b})$ . Note that if  $\text{id}^* \notin \mathcal{S}_i$ , then the two games  $\text{Game}_{3,i-1,2}$  and  $\text{Game}_{3,i-1,3}$  are identical. Thus we focus on the case  $\text{id}^* \in \mathcal{S}_i$ . In that case we show that  $\text{Game}_{3,i-1,3}$  is also identically distributed to  $\text{Game}_{3,i-1,2}$  using a statistical argument, which relies on the fact that vectors  $\mathbf{P}(\text{id}_j)^\top \mathbf{b}$  and  $\mathbf{P}(\text{id}_j)^\top \mathbf{d}$  are statistically independent since  $\mathbf{b}$  and  $\mathbf{d}$  are linearly independent. The same holds with respect to the matrix  $\mathbf{P}(0)$ . Moreover, since  $\text{id}^* \in \mathcal{S}_i$ , the set of values  $\{\mathbf{P}(\text{id}_j)\}_{\text{id}_j \in \mathcal{S}_i}, \mathbf{P}(\text{id}^*)\}$  are statistically independent from the value  $\mathbf{P}(0)$ —recall that the polynomial  $P$  is of degree  $t$ ; we are using Fact 1 from Sect. 2.2. Combining these two facts, we know that the vector  $\mathbf{P}(0)^\top \mathbf{d}$  is uniformly random, independent from everything else (challenge ciphertext, public key and other functional secret keys). Thus, it can act as a one-time pad on the value  $\mathbf{a}^\perp \mathbf{x}_\beta^\top \mathbf{y}$  that we wish to remove.

$\text{Game}_{3,i-1,4}$  : is defined as  $\text{Game}_{3,i-1,3}$ , except the vector  $\mathbf{d}$  used to compute the  $i$ 'th queried functional secret key is sampled  $\mathbf{d} \leftarrow_R \mathbb{Z}_p^2$ , instead of uniformly random over  $\mathbb{Z}_p^2 \setminus \text{Span}(\mathbf{b})$ . This is the reverse to the transition from  $\text{Game}_{3,i-1,1}$  to  $\text{Game}_{3,i-1,2}$ . By the same statistical argument, we obtain  $\text{Game}_{3,i-1,3} \approx_s \text{Game}_{3,i-1,4}$ .

Finally, note that  $\text{Game}_{3,i-1,4}$  is the same as  $\text{Game}_{3,i}$  except the  $i$ 'th queried key is computed using  $\llbracket \mathbf{d} \rrbracket_2 \leftarrow_R \mathbb{G}_2^2$  in the former, and  $\llbracket \mathbf{b} s_i \rrbracket_2 \in \mathbb{G}_2^2$  with  $s_i \leftarrow_R \mathbb{Z}_p$  in the latter. Therefore, we have  $\text{Game}_{3,i-1,4} \approx_c \text{Game}_{3,i}$  by the DDH assumption, which states that  $(\llbracket \mathbf{b} \rrbracket_2, \llbracket \mathbf{d} \rrbracket_2) \approx_c (\llbracket \mathbf{b} \rrbracket_2, \llbracket \mathbf{b} s_i \rrbracket_2)$  where  $\mathbf{b}, \mathbf{d} \leftarrow_R \mathbb{Z}_p^2, s_i \leftarrow_R \mathbb{Z}_p$ . The former distribution corresponds to  $\text{Game}_{3,i-1,4}$ , whereas the latter distribution corresponds to  $\text{Game}_{3,i}$ . Note that this transition is exactly reverse to the transition from  $\text{Game}_{3,i-1}$  to  $\text{Game}_{3,i-1,1}$ . This concludes the proof that  $\text{Game}_{3,i-1} \approx_c \text{Game}_{3,i}$  and consequently, that  $\text{Game}_3 \approx_c \text{Game}_4$ .

Note that in  $\text{Game}_4$ , the only values that possibly reveal some information about the bit  $\beta$  is the set  $\{\mathbf{x}_\beta^\top \mathbf{y}_i\}$  for all queries  $(\llbracket \mathbf{y}_i \rrbracket_2, \mathcal{S}_i)$  such that  $\text{id}^* \notin \mathcal{S}_i$ . Since the adversary  $\mathcal{A}$  is admissible, we know that for all such values,  $\mathbf{x}_\beta^\top \mathbf{y}_i = \mathbf{x}_0^\top \mathbf{y}_i = \mathbf{x}_1^\top \mathbf{y}_i$ . In other words, these values do not depend on  $\beta$  and the advantage of  $\mathcal{A}$  is 0.  $\square$

### 3.3 Revocations with Arbitrary-Size Identity Sets

Our previous construction requires that the size of any identities set  $\mathcal{S}$  be exactly  $t$  (a pre-established system parameter).

A possible way to relax this limitation is to introduce dummy identities and use them as “fillers”, to extend an identity set until it reaches size  $t$ . Furthermore, in order to make the secret-key size proportional to the identity set  $\mathcal{S}$ , we could run different instances of the IPFE for different set-size bounds  $t_1, \dots, t_n$ . A secret-key for set  $\mathcal{S}$  would then be issued only with respect to the  $i$ -th IPFE instance, where  $t_i$  is the smallest such that  $|\mathcal{S}| \leq t_i$ . (Ciphertexts would need to be provided with respect to all IPFE instances). A natural and effective choice for the values of  $t_i$  is the set of powers of 2. That way, the ciphertext-size would be increased by a factor of  $\log_2$  of the global maximum identity set size. Note

that such factor is logairthmic in the security parameter. This technique has already been used in the literature and in particular in the context of ABE, e.g. by Ostrovsky et al. [OSW07, Section 3.3].

## 4 Generic Construction of MA-ABE from IPFE

We present a modular construction of MA-ABE for non-monotone access structures based on inner-product FE schemes. We show that the resulting MA-ABE is super selectively secure for static corruptions, provided the underlying FE are super selectively secure. The security is proven in the random oracle model.

GlobalSetup( $1^\lambda$ ) :

Generate a pairing group  $\mathcal{PG} = (p, \mathbb{G}_1, \mathbb{G}_2, P_1, P_2, \mathbb{G}_t, e) \leftarrow \text{GGen}(1^\lambda)$  and a hash functions  $H : \{0, 1\}^* \rightarrow \mathbb{G}_2^3$  and return  $\text{gp} := (\mathcal{PG}, H)$ .

AuthSetup(gp) :

Compute  $(\text{pk}_\Gamma, \text{msk}_\Gamma) \leftarrow \Gamma.\text{Setup}(1^\lambda, \mathcal{PG})$  and  $(\text{pk}_\Sigma, \text{msk}_\Sigma) \leftarrow \Sigma.\text{Setup}(1^\lambda, \mathcal{PG})$ . return  $\text{pk} = (\text{pk}_\Gamma, \text{pk}_\Sigma)$  and  $\text{sk} = (\text{msk}_\Gamma, \text{msk}_\Sigma)$ .

Enc( $(\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}, \rho : [\ell] \rightarrow \{0, 1\}^* \cup (\{\neg\} \cdot \{0, 1\}^*))$ ,  $\{\text{pk}_i\}_{i \in [\nu]}$ ) :

Sample  $s \leftarrow_R \mathbb{Z}_p$ , and  $\{s_j\}_{j \in [\ell]} \leftarrow \text{Share}(\mathbf{M}, s)$ ,  $\{u_j\}_{j \in [\ell]} \leftarrow \text{Share}(\mathbf{M}, 0)$ ,  $\mathbf{a} \leftarrow_R \mathbb{Z}_p^3$ . For all  $j \in [\ell]$ , parse  $\text{pk}_{\theta(j)} = (\text{pk}_{\Gamma, \theta(j)}, \text{pk}_{\Sigma, \theta(j)})$ , set  $\mathbf{x}_j = (s_j, u_j \cdot \mathbf{a}) \in \mathbb{Z}_p^4$ , then

- if  $\rho(j) = \text{att}_j$  where  $\text{att}_j \in \{0, 1\}^*$ , then  $\text{ct}_j \leftarrow \Gamma.\text{Enc}(\text{pk}_{\Gamma, \theta(j)}, \mathbf{x}_j, \text{att}_j)$ .
- if  $\rho(j) = \neg \text{att}_j$  where  $\text{att}_j \in \{0, 1\}^*$ , then  $\text{ct}_j \leftarrow \Sigma.\text{Enc}(\text{pk}_{\Sigma, \theta(j)}, \mathbf{x}_j, \text{att}_j)$ .

Return  $(\{\text{ct}_j\}_{j \in [\ell]}, \kappa := \llbracket s \rrbracket_t)$ .

KeyGen(pk, sk, gid, S) :

Parse  $\text{sk} = (\text{msk}_\Gamma, \text{msk}_\Sigma)$ . Compute  $H(\text{gid}) = \llbracket \mathbf{z}_{\text{gid}} \rrbracket_2$ ,  $\text{sk}_\Sigma \leftarrow \Sigma.\text{KeyGen}(\text{msk}_\Sigma, \llbracket \mathbf{1}, \mathbf{z}_{\text{gid}} \rrbracket_2, \mathcal{S})$ , for all  $\text{att}_j \in \mathcal{S}$ ,  $\text{sk}_{\Gamma, j} \leftarrow \Gamma.\text{KeyGen}(\text{msk}_\Gamma, \llbracket \mathbf{1}, \mathbf{z}_{\text{gid}} \rrbracket_2, \text{att}_j)$ . Return  $\text{sk}_{\text{gid}, \mathcal{S}} = (\text{sk}_\Sigma, (\text{sk}_{\Gamma, j})_{\text{att}_j \in \mathcal{S}})$ .

Dec(ct,  $\{\text{sk}_{\text{gid}, \mathcal{S}_i}\}_i$ ) :

Parse the ciphertext  $\text{ct} = \{\text{ct}_j\}_{j \in [\ell]}$  which contains the description of an access structure  $(\mathbf{M}, \rho)$ . Let  $\mathcal{S} = \cup_{i \in [\nu]} \mathcal{S}_i$ . Compute  $\{\omega_j, \omega'_j\}_{j \in [\ell]}$  such that  $\sum_{\rho(j) \in \mathcal{S}} \omega_j \mathbf{M}_j + \sum_{\rho(j) = \neg \text{att}, \text{att} \notin \mathcal{S}} \omega'_j \mathbf{M}_j = \mathbf{1}$ . Return  $\sum_{\rho(j) \in \mathcal{S}} \omega_j \Gamma.\text{Dec}(\text{pk}_{\theta(j)}, \text{ct}_j, \text{sk}_{\text{gid}, \mathcal{S}_{\theta(j)}}) + \sum_{\rho(j) = \neg \text{att}, \text{att} \notin \mathcal{S}} \omega'_j \Sigma.\text{Dec}(\text{pk}_{\theta(j)}, \text{ct}_j, \text{sk}_{\text{gid}, \mathcal{S}_{\theta(j)}}) + \llbracket \mathbf{1}, \mathbf{z}_{\text{gid}} \rrbracket_2$ , where  $\llbracket \mathbf{z}_{\text{gid}} \rrbracket_2 = H(\text{gid})$ .

**Fig. 3.** Construction of Multi-Authority ABE from an ID-IPFE scheme  $\Gamma$  and an IPFE with revocations  $\Sigma$  (for vectors of dimension 4). Recall that  $\theta$  maps a row  $j \in [\ell]$  to the authority that owns the attribute associated to that row.

**Correctness.** Let  $\llbracket \mathbf{z}_{\text{gid}} \rrbracket_2 := H(\text{gid})$ . Observe that, by the correctness of  $\Gamma$  and  $\Sigma$ , we have:

$$\begin{aligned} & \sum_{\rho(j) \in S} \omega_j \Gamma.\text{Dec}(\text{pk}_{\theta(j)}, \text{ct}_j, \text{sk}_{\text{gid}, S_{\theta(j)}}) \llbracket 1, \mathbf{z}_{\text{gid}} \rrbracket_2 \\ & + \sum_{\rho(j) = \neg \text{att}, \text{att} \notin S} \omega'_j \Sigma.\text{Dec}(\text{pk}_{\theta(j)}, \text{ct}_j, \text{sk}_{\text{gid}, S_{\theta(j)}}) \llbracket 1, \mathbf{z}_{\text{gid}} \rrbracket_2 \\ & = \sum_{\rho(j) \in S} \omega_j \llbracket s_j + \mathbf{a}^\top \mathbf{z}_{\text{gid}} u_j \rrbracket_t + \sum_{\rho(j) = \neg \text{att}, \text{att} \notin S} \omega'_j \llbracket s_j + \mathbf{a}^\top \mathbf{z}_{\text{gid}} u_j \rrbracket_t \\ & = \llbracket s + \mathbf{a}^\top \mathbf{z}_{\text{gid}} \cdot 0 \rrbracket_t = \kappa . \end{aligned}$$

**Theorem 2 (Super-selective security).** *The scheme from Fig. 3, is a super-selectively secure MA-ABE with static corruption in the random oracle model, assuming the schemes  $\Gamma$  and  $\Sigma$  are super-selectively secure and the DDH assumption holds in  $\mathbb{G}_2$ .*

Combining with the existence of an ID-IPFE selectively secure under SXDH (from [ACGU20]) and Theorem 1 (the existence of selectively secure IPFE with revocations from SXDH) and noting that selective security implies super-selective security, we obtain the following corollary.

**Corollary 1.** *There exists a super-selectively secure MA-ABE with static corruptions from SXDH.*

We now proceed to prove the theorem.

*Proof.* We prove security via a sequence of hybrid games. We highlight in red the changes from one hybrid to the next when relevant.

**Game<sub>0</sub>** : The first game corresponds to the super-selective security game for MA-ABE with static corruptions, defined in Sect. 2.6. We recall it here for completeness. We call  $\mathcal{A}$  the admissible adversary. First,  $\mathcal{A}$  receives the global parameters  $\text{gp} = (\Gamma.\text{gp}, H)$ . Then, it can query its oracle  $\mathcal{O}_{\text{create}}$  that creates a new (honest) authority with an associated  $(\text{pk}, \text{sk})$  pair when invoked, adds  $\text{pk}$  to the set of honest authorities denoted by  $\mathcal{S}_{\text{hon}}$  and returns  $\text{pk}$  to  $\mathcal{A}$ . Then,  $\mathcal{A}$  sends  $(\mathbf{M}, \rho, \Pi_{\text{hon}}, \Pi_{\text{corr}})$  to its challenger, where  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$ ,  $\rho : [\ell] \rightarrow \mathbb{Z}_p$  is an access structure with attributes owned by the authorities in the set  $\Pi = \Pi_{\text{hon}} \cup \Pi_{\text{corr}}$ . Here,  $\Pi_{\text{hon}}$  is a set of honest authorities' public keys, that is,  $\Pi_{\text{hon}} \subseteq \mathcal{S}_{\text{hon}}$ , and  $\Pi_{\text{corr}}$  is a set of authorities' public key created by  $\mathcal{A}$  itself (and not via  $\mathcal{O}_{\text{create}}$ ). Because  $\mathcal{A}$  is free to create these public keys however it wants (potentially maliciously), these are referred to as corrupted authorities. Note that  $\mathcal{A}$  cannot query its oracle  $\mathcal{O}_{\text{corr}}$ , since we assume only static corruptions here. We write  $\Pi = \{\text{pk}_1, \dots, \text{pk}_\nu\}$ , and we define  $\theta : [\ell] \rightarrow [\nu]$ , which maps each column  $j \in [\ell]$  to the authority that owns the attribute associated with that column.

Afterwards, the adversary can query its oracle  $\mathcal{O}_{\text{KeyGen}}$  on inputs  $\text{pk} \in \mathcal{S}_{\text{hon}}$  associated with the secret key  $\text{sk} = (\text{msk}_\Sigma, \text{msk}_\Gamma)$  and  $\mathcal{S} \subset \mathcal{U}_{\text{pk}}$ , which computes  $\text{sk}_\Sigma \leftarrow \Sigma.\text{KeyGen}(\text{msk}_\Sigma, \llbracket 1, \mathbf{z}_{\text{gid}} \rrbracket_2, \mathcal{S})$  and for all attributes  $\text{att}_j \in \mathcal{S}$ , it

computes  $\text{sk}_{\Gamma,j} \leftarrow \Gamma.\text{KeyGen}(\text{msk}_{\Gamma}, \llbracket 1, z_{\text{gid}} \rrbracket_2, \text{att}_j)$ , where  $\llbracket z_{\text{gid}} \rrbracket_2 = H(\text{gid})$ . It returns  $\text{sk}_{\text{gid},\mathcal{S}} = (\text{sk}_{\Sigma}, (\text{sk}_{\Gamma,j})_{\text{att}_j \in \mathcal{S}})$  to the adversary  $\mathcal{A}$ .

At this point, the challenger samples  $s \leftarrow_R \mathbb{Z}_p$  and computes  $(s_1, \dots, s_{\ell}) \leftarrow \text{Share}(\mathbf{M}, s)$ ,  $(u_1, \dots, u_{\ell}) \leftarrow \text{Share}(\mathbf{M}, 0)$ <sup>2</sup>,  $\mathbf{a} \leftarrow_R \mathbb{Z}_p^3$ ,  $\kappa_0 = \llbracket s \rrbracket_{\mathbf{t}}$ ,  $\kappa_1 \leftarrow_R \mathbb{G}_{\mathbf{t}}$ ,  $\beta \leftarrow_R \{0, 1\}$ , for all  $j \in [\ell]$ ,  $\mathbf{x}_j = (s_j, u_j \cdot \mathbf{a}) \in \mathbb{Z}_p^4$ , and

- if  $\rho(j) = \text{att}_j$  where  $\text{att}_j \in \{0, 1\}^*$ , then  $\text{ct}_j \leftarrow \Gamma.\text{Enc}(\text{pk}_{\Gamma, \theta(j)}, \mathbf{x}_i, \rho(j))$ ,
- if  $\rho(j) = \neg \text{att}_j$  where  $\text{att}_j \in \{0, 1\}^*$ , then  $\text{ct}_j \leftarrow \Sigma.\text{Enc}(\text{pk}_{\Sigma, \theta(j)}, \mathbf{x}_i, \rho(j))$ .

It sets  $\text{ct}^* = \{\text{ct}_j\}_{j \in [\ell]}$  and returns  $(\text{ct}^*, \kappa_{\beta})$  to  $\mathcal{A}$ . Finally,  $\mathcal{A}$  outputs a guess  $\beta' \in \{0, 1\}$ . Recall that  $\mathcal{A}$  is admissible, which means it cannot compute  $\kappa_0$  from  $\text{ct}^*$  simply by correctness of the scheme with the user secret keys it queried and the secret key of the corrupted authorities (see Sect. 2.6 for more details). The experiment outputs 1 if  $\beta = \beta'$ , 0 otherwise.

In the following hybrids, we use the following dual basis: first, we choose a random basis  $(\mathbf{a}_1 | \mathbf{a}_2 | \mathbf{a}_3) \in \mathbb{Z}_p^{3 \times 3}$  of  $\mathbb{Z}_p^3$  such that  $\mathbf{a} = r_1 \mathbf{a}_1$  for  $r_1 \leftarrow_R \mathbb{Z}_p^*$  (recall that the vector  $\mathbf{a}$  is sampled to produce the challenge ciphertext). Strictly speaking, such a basis exists only when  $\mathbf{a} \neq \mathbf{0}$ . Since  $\mathbf{a}$  is sampled uniformly at random over  $\mathbb{Z}_p^3$ , it is different from  $\mathbf{0}$  with overwhelming probability. Thus, we implicitly assume  $\mathbf{a}$  is sampled uniformly over  $\mathbb{Z}_p^3 \setminus \{\mathbf{0}\}$  in the proof (this only changes the distribution by a negligible statistical distance). Then, we denote by  $(\mathbf{a}_1^* | \mathbf{a}_2^* | \mathbf{a}_3^*) \in \mathbb{Z}_p^{3 \times 3}$  its dual basis, that is, such that for all  $i, j \in \{1, 2, 3\}$ ,  $\mathbf{a}_i^{\top} \mathbf{a}_j^* = 0$  if  $i \neq j$  and  $\mathbf{a}_i^{\top} \mathbf{a}_j^* = 1$  if  $i = j$ . We make use of the following assumptions relative to the pairing groups  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$  and the random dual basis  $(\mathbf{a}_1 | \mathbf{a}_2 | \mathbf{a}_3)$  and  $(\mathbf{a}_1^* | \mathbf{a}_2^* | \mathbf{a}_3^*)$ .

**Assumption 1.**  $\{\mathbf{v} \leftarrow_R \mathbb{Z}_p^3 : (\llbracket \mathbf{a}_1 \rrbracket_1, \llbracket \mathbf{v} \rrbracket_2)\} \approx_c \{\mathbf{v} \leftarrow_R \text{Span}(\mathbf{a}_1^*) : (\llbracket \mathbf{a}_1 \rrbracket_1, \llbracket \mathbf{v} \rrbracket_2)\}$ .

This assumption is known to be implied by the DDH assumption in  $\mathbb{G}_2$  (see for instance [Lew12]).

Game<sub>1</sub>: is the same as Game<sub>0</sub> except that the outputs of the hash function are computed as follows: for all  $\text{gid}$ ,  $H(\text{gid}) = \llbracket z_{\text{gid}} \rrbracket_2$  where  $z_{\text{gid}} \leftarrow_R \text{Span}(\mathbf{a}_1^*)$ . We have Game<sub>0</sub>  $\approx_c$  Game<sub>1</sub> by **Assumption 1**. Technically, we need to use this assumption for each query of  $\mathcal{A}$  to the hash function  $H$  (modeled as a random oracle) using a hybrid argument.

Game<sub>2</sub>: is the same as Game<sub>1</sub>, except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + v_j \mathbf{a}_3), \rho(j)$ , for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $v_j = (\gamma, \mathbf{v})^{\top} \mathbf{M}_j$ ,  $\mathbf{v} \leftarrow_R \mathbb{Z}_p^{n-1}$ , and  $\gamma \leftarrow_R \mathbb{Z}_p$ . That is, the  $v_j$  are shares of a random value  $\gamma$ . Recall that  $\mathbf{a}_1, \mathbf{a}_3 \in \mathbb{Z}_p^3$  are vectors part of the basis  $(\mathbf{a}_1 | \mathbf{a}_2 | \mathbf{a}_3)$

<sup>2</sup> See Fig. 1 for the definition of the algorithm Share.

and  $\mathbf{a} = r_1 \mathbf{a}_1$  where  $r_1 \leftarrow_R \mathbb{Z}_p^*$ . The shares  $s_j$  and  $u_j$  are computed as before. For all  $j \in [\ell]$  such that  $\theta(j) \in \Pi_{\text{corr}}$ , the vector  $\mathbf{x}_j$  are as before. The challenge ciphertext is set to be  $(\{\text{ct}_j\}_{j \in [\ell]}, \kappa_\beta)$ , where  $\kappa_\beta$  is computed as before. We argue that  $\text{Game}_1 \approx_c \text{Game}_2$  using the super-selective security of  $\Gamma$  and  $\Sigma$ , since the extra red vector  $(0, v_j \mathbf{a}_3)$  is orthogonal to the vectors  $\llbracket 1, \mathbf{z}_{\text{gid}} \rrbracket_2$  from the user secret keys. This is because for all queried  $\text{gid}$ ,  $\mathbf{z}_{\text{gid}} \in \text{Span}(\mathbf{a}_1^*)$  and  $\mathbf{a}_3^\top \mathbf{a}_1^* = 0$ .

Game<sub>3</sub>: is the same as  $\text{Game}_2$ , except that the outputs of the hash function are computed as follows: for all  $\text{gid}$ ,  $H(\text{gid}) = \llbracket \mathbf{a}_1^* r_{\text{gid}} + \mathbf{a}_3^* \rrbracket_2$ , where  $r_{\text{gid}} \leftarrow_R \mathbb{Z}_p$ . We prove that  $\text{Game}_2 \approx_c \text{Game}_3$  in Lemma 2.

Game<sub>4</sub>: is the same as  $\text{Game}_3$ , except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s'_j, u_j r_1 \mathbf{a}_1 + v'_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $s'_j = (s + \gamma, \mathbf{w})^\top \mathbf{M}_j$  and  $v'_j = (0, \mathbf{v})^\top \mathbf{M}_j$ . That is, the  $s'_j$  are now shares of  $s + \gamma$  instead of  $s$  and the  $v'_j$  are now shares of  $\gamma$ . The shares  $u_j$  are computed as before. We argue that  $\text{Game}_3 \approx_c \text{Game}_4$  thanks to the super-selective security of  $\Gamma$  and  $\Sigma$ . Indeed, for all  $j \in [\ell]$  and all queried  $\text{gid}$ , we have  $(s'_j, u_j r_1 \mathbf{a}_1 + v'_j \mathbf{a}_3)^\top (1, \mathbf{a}_1^* r_{\text{gid}} + \mathbf{a}_3^*) = (s + \gamma, \mathbf{w})^\top \mathbf{M}_j + r_1 r_{\text{gid}} (0, \mathbf{u})^\top \mathbf{M}_j + (0, \mathbf{v})^\top \mathbf{M}_j = (s, \mathbf{w})^\top \mathbf{M}_j + r_1 r_{\text{gid}} (0, \mathbf{u})^\top \mathbf{M}_j + (\gamma, \mathbf{v})^\top \mathbf{M}_j = s_j + r_1 r_{\text{gid}} u_j + v_j = (s_j, u_j r_1 \mathbf{a}_1 + v_j \mathbf{a}_3)^\top (1, \mathbf{a}_1^* r_{\text{gid}} + \mathbf{a}_3^*)$ , just as in  $\text{Game}_3$ . That is, the change of the vectors encrypted under  $\Gamma$  from  $\text{Game}_3$  to  $\text{Game}_4$  preserves the value of the inner product.

Finally, to conclude the proof, we show that in  $\text{Game}_4$ , the advantage of  $\mathcal{A}$  is 0. This comes from the fact that the value  $\kappa_0 = \llbracket s \rrbracket_t$  is uniformly random, independent of the rest of the adversary's view. Indeed, the only place where the value  $s$  appears is in the challenge ciphertext, in the vectors  $\mathbf{x}_j$  encrypted under  $\Gamma$  or  $\Sigma$ . For all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , the vector  $\mathbf{x}_j$  is of the form  $\mathbf{x}_j = (s'_j, u_j r_1 \mathbf{a}_1 + v'_j \mathbf{a}_3)$  where  $s'_j$  is of the form  $s'_j = (s + \gamma, \mathbf{w})^\top \mathbf{M}_j$  for all  $j \in [\ell]$ . That is, the values  $s'_j$  are shares of the secret  $s + \gamma$ . But the value  $\gamma \leftarrow_R \mathbb{Z}_p$  is independent of the rest of the adversary's view, thus it acts as a one-time pad on  $s$ . Consequently,  $\mathbf{x}_j$  is independent of the value  $s$ . For all  $j \in [\ell]$  such that  $\theta(j) \in \Pi_{\text{corr}}$ , we have  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + v_j \mathbf{a}_3)$ , where the values  $s_j$  are shares of the secret  $s$ . But because the adversary  $\mathcal{A}$  is admissible, we know that the shares  $\{s_j\}_{j \in [\ell], \theta(j) \in \Pi_{\text{corr}}}$  are independent of  $s$ , by security of the MSP. Thus, both  $\kappa_0$  and  $\kappa_1$  are uniformly random independent of everything else, the view of the adversary does not depend on the bit  $\beta$ ; its advantage is 0.  $\square$

Now we state and prove the lemma used in the proof above. Its proof relies on the assumptions below, which are known to be implied by DDH in  $\mathbb{G}_2$  (see for instance [Lew12]).

**Lemma 2.** *We have  $\text{Game}_2 \approx_c \text{Game}_3$  assuming the super-selective security of  $\Gamma$  and  $\Sigma$ , and the SXDH assumption.*

To prove the lemma, we rely on the following assumptions, which are known to be implied by DDH in  $\mathbb{G}_2$  (see for instance [Lew12]).

**Assumption 2.**

$$\{v \leftarrow_R \text{Span}(\mathbf{a}_1^*), r_1, r_2 \leftarrow_R \mathbb{Z}_p^* : ([r_1 \mathbf{a}_1 + r_2 \mathbf{a}_2]_1, [\mathbf{a}_1]_1, [\mathbf{a}_3]_1, [\mathbf{a}_1^*]_2, [\mathbf{a}_3^*]_2, [v]_2)\} \\ \approx_c \{v \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*), r_1, r_2 \leftarrow_R \mathbb{Z}_p^* : ([r_1 \mathbf{a}_1 + r_2 \mathbf{a}_2]_1, [\mathbf{a}_1]_1, [\mathbf{a}_3]_1, [\mathbf{a}_1^*]_2, [\mathbf{a}_3^*]_2, [v]_2)\}.$$

**Assumption 3.**

$$\{v \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*), r \leftarrow_R \mathbb{Z}_p : ([\mathbf{a}_1]_1, [r \mathbf{a}_2 + \mathbf{a}_3]_1, [\mathbf{a}_1^*]_2, [\mathbf{a}_3^*]_2, [v]_2)\} \\ \approx_c \{v \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*), r \leftarrow_R \mathbb{Z}_p : ([\mathbf{a}_1]_1, [r \mathbf{a}_2 + \mathbf{a}_3]_1, [\mathbf{a}_1^*]_2, [\mathbf{a}_3^*]_2, [v]_2)\}.$$

*Proof.* To prove the lemma, we introduce the following hybrid games for all  $i \in \{0, \dots, q\}$  where  $q \in \mathbb{N}$  denotes the number of distinct  $\text{gid}$  queried via  $\mathcal{O}_{\text{KeyGen}}$ :  $\text{Game}_{2,i}$  is like  $\text{Game}_2$ , except that for the first  $i$ 'th  $\text{gid}$ ,  $\mathcal{O}_{\text{KeyGen}}$  behaves like in  $\text{Game}_3$ . Namely, for the first  $i$ 'th  $\text{gid}$  queried to  $\mathcal{O}_{\text{KeyGen}}$ , the oracle uses  $H(\text{gid}) = [\mathbf{a}_1^* r_{\text{gid}} + \mathbf{a}_3^*]_2$ , whereas it uses  $H(\text{gid}) = [\mathbf{a}_1^* r_{\text{gid}}]_2$  for the last  $q - i$  queries. It is clear by definition of the games that  $\text{Game}_{2,0} = \text{Game}_2$  and  $\text{Game}_{2,q} = \text{Game}_3$ . We prove that for all  $i \in [q]$ ,  $\text{Game}_{2,i-1} \approx_c \text{Game}_{2,i}$ . To do so, we use the following hybrid games.

$\underline{\text{Game}_{2,i-1,1}}$  : is the same as  $\text{Game}_{2,i-1}$ , except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + u_j r_2 \mathbf{a}_2 + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $r_2 \leftarrow_R \mathbb{Z}_p^*$ . We argue that  $\text{Game}_{2,i-1} \approx_c \text{Game}_{2,i-1,1}$  thanks to the super-selective security of  $\Gamma$  and  $\Sigma$ . Indeed, for all  $j \in [\ell]$  and all queried  $\text{gid}$ , we have  $\mathbf{z}_{\text{gid}} \in \text{Span}(\mathbf{a}_1^*, \mathbf{a}_3^*)$ , thus  $(s_j, u_j r_1 \mathbf{a}_1 + u_j r_2 \mathbf{a}_2 + v_j \cdot \mathbf{a}_3)^\top (1, \mathbf{z}_{\text{gid}}) = (s_j, u_j r_1 \mathbf{a}_1 + v_j \cdot \mathbf{a}_3)^\top (1, \mathbf{z}_{\text{gid}})$ , just as in game  $\text{Game}_{2,i-1}$ , since  $\mathbf{a}_2^\top \mathbf{a}_1^* = \mathbf{a}_2^\top \mathbf{a}_3^* = 0$ .

$\underline{\text{Game}_{2,i-1,2}}$  : is the same as  $\text{Game}_{2,i-1,1}$  except that the output of the hash function on the  $i$ 'th queried global identifier, which we denote by  $\text{gid}_i$ , is computed as follows:  $H(\text{gid}_i) = [\mathbf{z}_{\text{gid}_i}]_2$  where  $\mathbf{z}_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*)$ , as opposed to uniformly random over  $\text{Span}(\mathbf{a}_1^*)$  in  $\text{Game}_{2,i-1,1}$ . We have  $\text{Game}_{2,i-1,1} \approx_c \text{Game}_{2,i-1,2}$  by **Assumption 2**.

$\underline{\text{Game}_{2,i-1,3}}$  : is the same as  $\text{Game}_{2,i-1,2}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + r_j \mathbf{a}_2 + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $r_j = (0, \mathbf{r})^\top \mathbf{M}_j$ , and  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$ . We have that  $\text{Game}_{2,i-1,2} \approx_c \text{Game}_{2,i-1,3}$  from the DDH assumption in  $\mathbb{G}_1$ , which implies that  $\{r_2 \leftarrow_R \mathbb{Z}_p^*, \mathbf{u} \leftarrow_R \mathbb{Z}_p^{n-1} : ([\mathbf{u}]_1, [r_2 \mathbf{u}]_1)\} \approx_c \{\mathbf{u}, \mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1} : ([\mathbf{u}]_1, [\mathbf{r}]_1)\}$ <sup>3</sup>

$\underline{\text{Game}_{2,i-1,4}}$  : is the same as  $\text{Game}_{2,i-1,3}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + r_j \mathbf{a}_2 + \eta_j \mathbf{a}_2 + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where the value  $\eta_j$  is defined as  $\eta(1, \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j$ , where  $\eta \leftarrow_R \mathbb{Z}_p$  and  $\mathbf{w}_{\text{gid}_i}$  is a vector such that  $(1, \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j = 0$  for all  $j \in [\ell]$  such that

<sup>3</sup> Strictly speaking, the DDH as per Definition 3 is stated with  $r_2 \leftarrow_R \mathbb{Z}_p$ , not  $r_2 \leftarrow_R \mathbb{Z}_p^*$  used here. This makes no difference, however, since the two distributions are within negligible statistical distance.



$\rho(j) \in \mathcal{S}_{\text{gid}_i}$  or  $(\rho(j) = \neg \text{att}_j$  and  $\text{att}_j \in \{0, 1\}^* \setminus \mathcal{S}_{\text{gid}_i})$ . The set  $\mathcal{S}_{\text{gid}_i}$  is defined as  $\mathcal{S}_{\text{gid}_i} = \cup_{\text{pk} \in \mathcal{S}_{\text{hon}}, (\text{pk}, \text{gid}_i, \mathcal{S}) \in \mathcal{Q}_{\text{KeyGen}}} \mathcal{S}$ . We know that  $\mathcal{S}_{\text{gid}_i}$  does not satisfy the access structure  $(\mathbf{M}, \rho)$  of the challenge ciphertext, because the adversary is admissible. Thus, by security of the access structure (Lemma 1), we know that such a vector  $\mathbf{w}_{\text{gid}_i} \in \mathbb{Z}_p^{n-1}$  exists. Note that we crucially rely on the selectivity here, since the vector  $\mathbf{w}_{\text{gid}_i}$  used in the challenge ciphertext depends on attributes queried to  $\mathcal{O}_{\text{KeyGen}}$ . The fact that  $\text{Game}_{2,i-1.4} \approx_c \text{Game}_{2,i-1.3}$  follows from the super-selective security of  $\Gamma$  and  $\Sigma$ . Indeed, the extra red component  $\eta_j \mathbf{a}_2$  encrypted under  $\Gamma$  or  $\Sigma$  never interacts with the vectors used to produce user secret keys. Namely, for all  $\text{gid} \neq \text{gid}_i$ , we have  $H(\text{gid}) = \llbracket \mathbf{z}_{\text{gid}} \rrbracket_2$  with  $\mathbf{z}_{\text{gid}} \in \text{Span}(\mathbf{a}_1^*, \mathbf{a}_3^*)$  so  $(0, \eta_j \mathbf{a}_2)^\top (1, \mathbf{z}_{\text{gid}}) = 0$ . For  $\text{gid} = \text{gid}_i$ , we argue that for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , either  $\eta_j = 0$ , or the extra  $\eta_j \mathbf{a}_2$  can be added thanks to the super-selective security of  $\Gamma$  and  $\Sigma$ . When  $\rho(j) \in \mathcal{S}_{\text{gid}_i}$  or  $\rho(j) = \neg \text{att}$  with  $\text{att} \in \{0, 1\}^* \setminus \mathcal{S}_{\text{gid}_i}$ , we know that  $\eta_j = 0$ . When  $\rho(j)$  is not of this form, then we know that none of the functional secret keys generated by  $\mathcal{O}_{\text{KeyGen}}$  on  $\text{gid}_i$  decrypt the ciphertext  $\text{ct}_j$ . Thus, we can conclude using the super-selective security of  $\Sigma$  and  $\Gamma$ .

$\text{Game}_{2,i-1.5}$  : is the same as  $\text{Game}_{2,i-1.4}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + \eta'_j \mathbf{a}_2 + v_j \mathbf{a}_3)$  where  $\eta'_j = (\eta, \mathbf{r})^\top \mathbf{M}_j$ , for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ . The fact that  $\text{Game}_{2,i-1.5} = \text{Game}_{2,i-1.4}$  follows from the fact a uniformly random vector  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$  is distributed identically to an offset  $\mathbf{x} \in \mathbb{Z}_p^{n-1}$  plus a uniformly random vector  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$ . This is true no matter the value of  $\mathbf{x}$ , as long as  $\mathbf{r}$  is sampled independently of  $\mathbf{x}$ . So, the following distributions are equal:  $\{r_j + \eta_j\}_{j \in [\ell]} = \{(0, \mathbf{r})^\top \mathbf{M}_j + \eta(1, \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j\}_{j \in [\ell]} = \{(\eta, \mathbf{r} + \eta \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j\}_{j \in [\ell]} \equiv \{(\eta, \mathbf{r})^\top \mathbf{M}_j\}_{j \in [\ell]} = \{\eta'_j\}_{j \in [\ell]}$ . This first distribution corresponds to  $\text{Game}_{2,i-1.4}$ , whereas the last distribution corresponds to  $\text{Game}_{2,i-1.5}$ .

$\text{Game}_{2,i-1.6}$  : is the same as  $\text{Game}_{2,i-1.5}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + r'_j \mathbf{a}_2 + v_j \mathbf{a}_3)$  or all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $r'_j = r(\gamma, \mathbf{v})^\top \mathbf{M}_j$ ,  $r \leftarrow_R \mathbb{Z}_p$ . Recall that  $\gamma \in \mathbb{Z}_p$  and  $\mathbf{v} \in \mathbb{Z}_p^{n-1}$  are used to compute the shares  $v_j$ , namely  $v_j = (\gamma, \mathbf{v})^\top \mathbf{M}_j$ . We argue that  $\text{Game}_{2,i-1.5} \approx_c \text{Game}_{2,i-1.6}$  using the DDH assumption in  $\mathbb{G}_1$ , which implies that  $\{\mathbf{r}, \mathbf{v} \leftarrow_R \mathbb{Z}_p^{n-1}, \eta, \gamma \leftarrow_R \mathbb{Z}_p : (\llbracket \eta \rrbracket_1, \llbracket \mathbf{r} \rrbracket_1, \llbracket \gamma \rrbracket_1, \llbracket \mathbf{v} \rrbracket_1)\} \approx_c \{\mathbf{v} \leftarrow_R \mathbb{Z}_p^{n-1}, r, \gamma \leftarrow_R \mathbb{Z}_p : (\llbracket r \gamma \rrbracket_1, \llbracket r \mathbf{v} \rrbracket_1, \llbracket \gamma \rrbracket_1, \llbracket \mathbf{v} \rrbracket_1)\}$ .

$\text{Game}_{2,i-1.7}$  : is the same as  $\text{Game}_{2,i-1.6}$  except that the output of the hash function on the  $i$ 'th queried global identifier, which we denote by  $\text{gid}_i$ , is computed as follows:  $H(\text{gid}_i) = \llbracket \mathbf{z}_{\text{gid}_i} + \mathbf{a}_3^* \rrbracket_2$  where  $\mathbf{z}_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*)$ . We have  $\text{Game}_{2,i-1.6} \approx_c \text{Game}_{2,i-1.7}$  by **Assumption 3**. Indeed, we have  $\{\mathbf{z}_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*) : \llbracket \mathbf{z}_{\text{gid}_i} \rrbracket_2\} \approx_c \{\mathbf{z}_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*) : \llbracket \mathbf{z}_{\text{gid}_i} \rrbracket_2\} \equiv \{\mathbf{z}_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*) : \llbracket \mathbf{z}_{\text{gid}_i} + \mathbf{a}_3^* \rrbracket_2\} \approx_c \{\mathbf{z}_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*) : \llbracket \mathbf{z}_{\text{gid}_i} + \mathbf{a}_3^* \rrbracket_2\}$ , where the  $\approx_c$  follows from **Assumption 3**. The first distribution corresponds to  $\text{Game}_{2,i-1.6}$ , whereas the last distribution corresponds to  $\text{Game}_{2,i-1.7}$ . Note

that for readability we omit the other values ( $\llbracket \mathbf{a}_1 \rrbracket_1, \llbracket r\mathbf{a}_2 + \mathbf{a}_3 \rrbracket_1, \llbracket \mathbf{a}_1^* \rrbracket_2, \llbracket \mathbf{a}_3^* \rrbracket_2$ ) present in the output of all distributions. These values are sufficient to generate the entire adversary's view.

Game $_{2,i-1,8}$ : is the same as Game $_{2,i-1,7}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + \eta'_j \mathbf{a}_2 + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $\eta'_j = (\eta, \mathbf{r})^\top \mathbf{M}_j$ ,  $\eta \leftarrow_R \mathbb{Z}_p$ ,  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$ . This is the reverse of the transition from Game $_{2,i-1,5}$  and Game $_{2,i-1,6}$ . We have Game $_{2,i-1,7} \approx_c \text{Game}_{2,i-1,8}$  using the DDH assumption in  $\mathbb{G}_1$ , which implies that  $\{r, \gamma \leftarrow_R \mathbb{Z}_p, \mathbf{v} \leftarrow_R \mathbb{Z}_p^{n-1} : (\llbracket r\gamma \rrbracket_1, \llbracket r\mathbf{v} \rrbracket_1, \llbracket \gamma \rrbracket_1, \llbracket \mathbf{v} \rrbracket_1)\} \approx_c \{\eta, \gamma \leftarrow_R \mathbb{Z}_p, \mathbf{r}, \mathbf{v} \leftarrow_R \mathbb{Z}_p^{n-1} : (\llbracket \eta \rrbracket_1, \llbracket \mathbf{r} \rrbracket_1, \llbracket \gamma \rrbracket_1, \llbracket \mathbf{v} \rrbracket_1)\}$ .

Game $_{2,i-1,9}$ : is the same as Game $_{2,i-1,8}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + r_j \mathbf{a}_2 + \eta_j + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $r_j = (0, \mathbf{r})^\top \mathbf{M}_j$ ,  $\eta_j = \eta(1, \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j$ ,  $\eta \leftarrow_R \mathbb{Z}_p$  and  $\mathbf{w}_{\text{gid}_i}$  is defined as before. This is the reverse of the transition from Game $_{2,i-1,4}$  and Game $_{2,i-1,5}$ . The fact that Game $_{2,i-1,8} = \text{Game}_{2,i-1,9}$  follows from the fact a uniformly random vector  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$  is distributed identically to an offset  $\mathbf{x} \in \mathbb{Z}_p^{n-1}$  plus a uniformly random vector  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$ , as long as  $\mathbf{r}$  is sampled independently of  $\mathbf{x}$ . So, the following distributions are equal:  $\{\eta'_j\}_{j \in [\ell]} = \{(\eta, \mathbf{r})^\top \mathbf{M}_j\}_{j \in [\ell]} \equiv \{(\eta, \mathbf{r} + \eta \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j\}_{j \in [\ell]} = \{(0, \mathbf{r})^\top \mathbf{M}_j + \eta(1, \mathbf{w}_{\text{gid}_i})^\top \mathbf{M}_j\}_{j \in [\ell]} = \{r_j + \eta_j\}_{j \in [\ell]}$ . This first distribution corresponds to Game $_{2,i-1,8}$ , whereas the last distribution corresponds to Game $_{2,i-1,9}$ .

Game $_{2,i-1,10}$ : is the same as Game $_{2,i-1,9}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + r_j \mathbf{a}_2 + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $r_j = (0, \mathbf{r})^\top \mathbf{M}_j$ ,  $\mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1}$ . This is the reverse of the transition from Game $_{2,i-1,3}$  and Game $_{2,i-1,4}$ . The fact that Game $_{2,i-1,9} \approx_c \text{Game}_{2,i-1,10}$  follows from the super-selective security of  $\Gamma$  and  $\Sigma$ . Indeed, the component  $\eta_j \mathbf{a}_2$  encrypted under  $\Gamma$  and  $\Sigma$  in Game $_{2,i-1,9}$  never interacts with the vectors used to produce user secret keys. Namely, for all  $\text{gid} \neq \text{gid}_i$ , we have  $H(\text{gid}) = \llbracket \mathbf{z}_{\text{gid}} \rrbracket_2$  with  $\mathbf{z}_{\text{gid}} \in \text{Span}(\mathbf{a}_1^*, \mathbf{a}_3^*)$  so  $(0, \eta_j \mathbf{a}_2)^\top (1, \mathbf{z}_{\text{gid}}) = 0$ . For  $\text{gid} = \text{gid}_i$ , we know that all queries  $(\text{pk}, \text{gid}_i, \mathcal{S})$  to  $\mathcal{O}_{\text{KeyGen}}$  are such that  $\mathcal{S} \in \mathcal{S}_{\text{gid}_i}$  (by definition of the set  $\mathcal{S}_{\text{gid}_i}$ ), and, as argued before, we know that for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , either  $\eta_j = 0$  or  $\text{ct}_j$  cannot be decrypted by the functional secret keys generated by  $\mathcal{O}_{\text{KeyGen}}$  on  $\text{gid}_i$ .

Game $_{2,i-1,11}$ : is the same as Game $_{2,i-1,10}$  except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + u_j r_2 \mathbf{a}_2 + v_j \cdot \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ , where  $r_2 \leftarrow_R \mathbb{Z}_p^*$ . This is the reverse of the transition from Game $_{2,i-1,2}$  and Game $_{2,i-1,3}$ . We have that Game $_{2,i-1,10} \approx_c \text{Game}_{2,i-1,11}$  from the DDH assumption in  $\mathbb{G}_1$ , which implies that  $\{\mathbf{u}, \mathbf{r} \leftarrow_R \mathbb{Z}_p^{n-1} : (\llbracket \mathbf{u} \rrbracket_1, \llbracket \mathbf{r} \rrbracket_1)\} \approx_c$

$\{r_2 \leftarrow_R \mathbb{Z}_p^*, \mathbf{u} \leftarrow_R \mathbb{Z}_p^{n-1} : ([\mathbf{u}]_1, [r_2 \cdot \mathbf{u}]_1)\}^4$ . This first distribution corresponds to  $\text{Game}_{2,i-1,10}$ , whereas the last distribution corresponds to  $\text{Game}_{2,i-1,11}$ .

$\text{Game}_{2,i-1,12}$  : is the same as  $\text{Game}_{2,i-1,11}$  except that the output of the hash function on the  $i$ 'th queried global identifier, which we denote by  $\text{gid}_i$ , is computed as follows:  $H(\text{gid}_i) = \llbracket z_{\text{gid}_i} + \mathbf{a}_3^* \rrbracket_2$  where  $z_{\text{gid}_i} \leftarrow_R \text{Span}(\mathbf{a}_1^*)$ , as opposed to uniformly random over  $\text{Span}(\mathbf{a}_1^*, \mathbf{a}_2^*)$  in  $\text{Game}_{2,i-1,11}$ . This is the reverse of the transition from  $\text{Game}_{2,i-1,1}$  and  $\text{Game}_{2,i-1,2}$ . We have  $\text{Game}_{2,i-1,1} \approx_c \text{Game}_{2,i-1,2}$  by **Assumption 2**.

$\text{Game}_{2,i}$  : is the same as  $\text{Game}_{2,i-1,12}$ , except that the challenge ciphertext uses the vectors  $\mathbf{x}_j = (s_j, u_j r_1 \mathbf{a}_1 + v_j \mathbf{a}_3)$  for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$ . That is, we remove the component  $u_j r_2 \mathbf{a}_2$ . We argue that  $\text{Game}_{2,i-1,12} \approx_c \text{Game}_{2,i}$  thanks to the super-selective security of  $\Gamma$  and  $\Sigma$ . Indeed, for all  $j \in [\ell]$  such that  $\theta(j) \in \mathcal{S}_{\text{hon}}$  and all queried  $\text{gid}$ , we have  $z_{\text{gid}} \in \text{Span}(\mathbf{a}_1^*, \mathbf{a}_3^*)$ , thus  $(s_j, u_j r_1 \mathbf{a}_1 + u_j r_2 \mathbf{a}_2 + v_j \cdot \mathbf{a}_3)^\top (1, z_{\text{gid}}) = (s_j, u_j r_1 \mathbf{a}_1 + v_j \cdot \mathbf{a}_3)^\top (1, z_{\text{gid}})$ , just as in game  $\text{Game}_{2,i}$ , since  $\mathbf{a}_2^\top \mathbf{a}_1^* = \mathbf{a}_2^\top \mathbf{a}_3^* = 0$ .  $\square$

## References

- [ACGU20] Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 467–497. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_16](https://doi.org/10.1007/978-3-030-64840-4_16)
- [AKW18] Agrawal, S., Koppula, V., Waters, B.: Impossibility of simulation secure functional encryption even with random oracles. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11239, pp. 659–688. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03807-6\\_24](https://doi.org/10.1007/978-3-030-03807-6_24)
- [ALS16] Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53015-3\\_12](https://doi.org/10.1007/978-3-662-53015-3_12)
- [AYY22] Agrawal, S., Yadav, A., Yamada, S.: Multi-input attribute based encryption and predicate encryption. In: Dodis, Y., Shrimpton, T. (eds) Advances in Cryptology – CRYPTO 2022. CRYPTO 2022. LNCS, vol. 13507, pp. 590–621. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-15802-5\\_21](https://doi.org/10.1007/978-3-031-15802-5_21)
- [Bei96] Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Ph.D, Technion - Israel Institute of Technology (1996)
- [BSW11] 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](https://doi.org/10.1007/978-3-642-19571-6_16)





<sup>4</sup> Again, strictly speaking, the DDH as per Definition 3 is stated with  $r_2 \leftarrow_R \mathbb{Z}_p$ , not  $r_2 \leftarrow_R \mathbb{Z}_p^*$  but as we argued above, this makes no difference since the two distributions are within negligible statistical distance.

- [CC09] Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009, pp. 121–130. ACM Press, November 2009
- [Cha07] Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-70936-7\\_28](https://doi.org/10.1007/978-3-540-70936-7_28)
- [CS02] 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](https://doi.org/10.1007/3-540-46035-7_4)
- [DKW21a] Datta, P., Komargodski, I., Waters, B.: Decentralized multi-authority ABE for DNFs from LWE. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 177–209. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77870-5\\_7](https://doi.org/10.1007/978-3-030-77870-5_7)
- [DKW21b] Datta, P., Komargodski, I., Waters, B.: Decentralized multi-authority ABE for  $\text{NC}^1$  from computational-BDH. Cryptology ePrint Archive (2021)
- [DP19] Dufour-Sans, E., Pointcheval, D.: Unbounded inner-product functional encryption with succinct keys. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 426–441. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-21568-2\\_21](https://doi.org/10.1007/978-3-030-21568-2_21)
- [GPSW06] 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 (2006). Available as Cryptology ePrint Archive Report 2006/309
- [Kim19] Kim, S.: Multi-authority attribute-based encryption from LWE in the OT model. IACR Cryptology ePrint Archive 2019:280 (2019)
- [KW93] Karchmer, M., Wigderson, A.: On span programs. In: Structure in Complexity Theory Conference, 1993, Proceedings of the Eighth Annual, pp. 102–111, May 1993
- [LCLS08] Lin, H., Cao, Z., Liang, X., Shao, J.: Secure threshold multi authority attribute based encryption without a central authority. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 426–436. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-89754-5\\_33](https://doi.org/10.1007/978-3-540-89754-5_33)
- [Lew12] Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_20](https://doi.org/10.1007/978-3-642-29011-4_20)
- [LW11] Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20465-4\\_31](https://doi.org/10.1007/978-3-642-20465-4_31)
- [MJ18] Michalevsky, Y., Joye, M.: Decentralized policy-hiding ABE with receiver privacy. In: Lopez, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018. LNCS, vol. 11099, pp. 548–567. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-98989-1\\_27](https://doi.org/10.1007/978-3-319-98989-1_27)
- [MKE08] Müller, S., Katzenbeisser, S., Eckert, C.: Distributed attribute-based encryption. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 20–36. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-00730-9\\_2](https://doi.org/10.1007/978-3-642-00730-9_2)

- [OSW07] Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007, pp. 195–203. ACM Press, October 2007
- [OT09] Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10366-7\\_13](https://doi.org/10.1007/978-3-642-10366-7_13)
- [OT13] Okamoto, T., Takashima, K.: Decentralized attribute-based signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 125–142. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36362-7\\_9](https://doi.org/10.1007/978-3-642-36362-7_9)
- [Rog15] Rogaway, P.: The moral character of cryptographic work. IACR Cryptology ePrint Archive 2015:1162 (2015)
- [RW15] Rouselakis, Y., Waters, B.: Efficient statically-secure large-universe multi-authority attribute-based encryption. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 315–332. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-47854-7\\_19](https://doi.org/10.1007/978-3-662-47854-7_19)
- [SW05] 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](https://doi.org/10.1007/11426639_27)
- [TT18] Tomida, J., Takashima, K.: Unbounded inner product functional encryption from bilinear maps. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 609–639. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03329-3\\_21](https://doi.org/10.1007/978-3-030-03329-3_21)
- [WFL19] Wang, Z., Fan, X., Liu, F.-H.: FE for inner products and its application to decentralized ABE. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 97–127. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17259-6\\_4](https://doi.org/10.1007/978-3-030-17259-6_4)
- [WWW22] Waters, B., Wee, H., Wu, D.J.: Multi-authority ABE from lattices without random oracles. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography Conference. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-22318-1\\_23](https://doi.org/10.1007/978-3-031-22318-1_23)



# Multi-instance Secure Public-Key Encryption

Carlo Brunetta<sup>1</sup> , Hans Heum<sup>2</sup>  , and Martijn Stam<sup>1</sup> 

<sup>1</sup> Simula UiB, Bergen, Norway  
{carlo, martijn}@simula.no

<sup>2</sup> Department of Mathematical Sciences, NTNU - Norwegian University of Science and Technology, Trondheim, Norway  
hans.heum@ntnu.no

**Abstract.** Mass surveillance targets many users at the same time with the goal of learning as much as possible. Intuitively, breaking many users' cryptography simultaneously should be at least as hard as that of only breaking a single one, but ideally security degradation is gradual: an adversary ought to work harder to break more. Bellare, Ristenpart and Tessaro (Crypto'12) introduced the notion of multi-instance security to capture the related concept for password hashing with salts. Auerbach, Giacon and Kiltz (Eurocrypt'20) motivated the study of public key encryption (PKE) in the multi-instance setting, yet their technical results are exclusively stated in terms of key encapsulation mechanisms (KEMs), leaving a considerable gap.

We investigate the multi-instance security of public key encryption. Our contributions are twofold. Firstly, we define and compare possible security notions for multi-instance PKE, where we include PKE schemes whose correctness is not perfect. Secondly, we observe that, in general, a hybrid encryption scheme of a multi-instance secure KEM and an arbitrary data encapsulation mechanism (DEM) is unlikely to inherit the KEM's multi-instance security. Yet, we show how with a suitable information-theoretic DEM, and a computationally secure key derivation function if need be, inheritance is possible. As far as we are aware, ours is the first inheritance result in the challenging multi-bit scenario.

**Keywords:** Multi-Instance Security · Hybrid Encryption · Property Inheritance · Mass Surveillance

## 1 Introduction

Security of cryptographic schemes is increasingly studied concretely. The question changes from whether a scheme is secure or not, to how secure it is. The change in emphasis also results in increased importance in more realistic security notions that model a world where an adversary might have many potential targets. If an adversary simply tries to learn something about one of its  $\kappa$  targets, then intuitively the more targets there are, the easier the adversary's job

---

Work by Hans Heum performed as part of his PhD studies at Simula UiB.

becomes. Indeed, using simple hybrid arguments results in a security degradation that is linear in  $\kappa$ . But what happens if the adversary is greedy and wants to learn more, maybe even targets everyone? On the one hand, one could argue that if breaking one instance is hard, then so is breaking many. Yet, on the other hand, one would hope that breaking multiple instances, say  $n$ , is strictly harder than breaking just a single one.

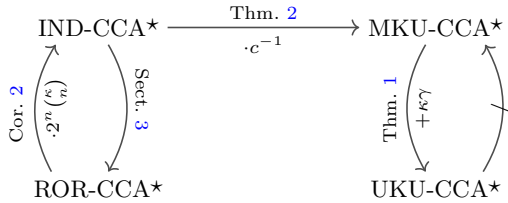
This second perspective made Bellare, Ristenpart and Tessaro [12], henceforth BRT, realize that new security notions are needed to reason about such greedy adversaries. They were motivated by how salts in password hashing protect against attackers re-using precomputation to retrieve multiple passwords. For their study into probabilistic symmetric schemes, they identified left-or-right indistinguishability under xor as the strongest notion. Roughly speaking, there are  $\kappa$  keys in the system each associated with its own left-or-right challenge bit  $b_i$  and the goal of the adversary is to guess the xor of all those bits.

Recently, Auerbach, Giacon and Kiltz [4], henceforth AGK, argued the importance of BRT's concept to protect against mass surveillance. They introduced the  $(n, \kappa)$  scaling factor as the effort to break  $n$  out of  $\kappa$  instances relative to the effort needed to break a single instance. After recalling several well-known greedy attacks against public key schemes with dubious scaling factors, they set out to provide an encryption scheme with good, non-trivial scaling factor.

They discussed various versions of Hashed ElGamal that differed in whether users shared group parameters and/or generators, plus whether the underlying group was elliptic curve or finite field based. In the programmable random oracle model, they showed that the multi-instance security of Hashed ElGamal tightly relates to a novel multi-instance Gap Computational Diffie–Hellman (MI-GapCDH) assumption, whose validity was further supported by an analysis in the generic group model.

There was, or rather is, just one small problem: Hashed ElGamal is a key encapsulation mechanism (KEM), not a public key encryption (PKE) scheme. Indeed, although AGK use PKE as their motivation, their formalization is entirely centred around KEMs. Of course, Cramer and Shoup [18] already showed how a secure KEM can be combined with a secure data encapsulation mechanism (DEM) to create a secure PKE (for various notions of security). This so-called hybrid encryption paradigm is widely deployed in the real world, yet, can its composition theorem be easily lifted to the multi-instance setting?

For key unrecoverability, all seems fine, but for indistinguishability one quickly uncovers various challenges. Consider an adversary  $\mathbb{A}$  that wants to recover  $n$  out of  $\kappa$  challenge bits  $b_i$ : it can attempt to recover roughly half of its  $b_i$  by somehow breaking the DEM, and recovering the remaining half by breaking the KEM. Intuitively, such a divide-and-conquer strategy essentially rules out inheriting full multi-instance security of both KEM and DEM simultaneously. Instead, perhaps we should aim to bound an adversary's multi-instance advantage against the hybrid encryption in terms of either breaking the full multi-instance security of the KEM or breaking only one of many instances of the DEM.



**Fig. 1.** An overview of multi-instance security notions for public-key encryption, where  $\gamma$  relates to imperfect correctness (Definition 1), and the loss factor  $c$  is explained in Theorem 2.

Special care would have to be taken to ensure that the corresponding multi-user DEM advantage is not overwhelming the multi-instance KEM advantage. After all, already when showing multi-user security of hybrid encryption, ensuring the DEM advantage does not overshadow the multi-user KEM advantage is challenging [23]. Furthermore, the study of multi-user KEMs highlights a second, more technical problem.

For multi-user security, there are essentially two different formalizations possible: one where each user comes with its own challenge bit and one where the users share a global challenge bit. Jager et al. [29] recently observed that only the latter lends itself to an easy adaptation of composition theorems using KEMs, as it allows a simple game-hop where all KEM-derived ephemeral keys are replaced by randomly selected keys (decoupled from the KEM encapsulations). That proof technique fails when there are multiple challenge bits. Unfortunately, for multi-instance security, the only option available is a notion with multiple challenge bits. In such a setting, inheritance of security properties of the KEM to any construction based on the KEM is an open problem.

**Our Contribution.** As mentioned above, multi-instance security was introduced by BRT in the context of probabilistic symmetric primitives and later adapted to key encapsulation mechanisms by AGK, who provide an excellent motivation for the study of multi-instance security in a public key setting. We adapt those notions to multi-instance security for PKE schemes, but make a number of non-trivial changes in the process. Firstly, we observe that the mechanisms used by BRT and AGK to model multi-instance games differ, which seems to have gone unnoticed hitherto. BRT’s mechanism is stronger as it allows for corruptions (denoted by  $\star$ ), yet AGK’s mechanism is more expressive by making explicit how many instances an adversary should break. We use elements of both in our notions, incorporating both BRT’s corruptions and AGK’s explicit expression of the number of targeted instances. Secondly, we allow for correctness to be imperfect, which has ramifications for how to deal with decryption oracles (for chosen-ciphertext attacks) and corruptions. We delve into the differences between the various mechanisms in Sect. 3.3, furthermore we use our revised mechanism to study a number of related notions, as summarized in Fig. 1.



In more detail, we start out by porting BRT’s notion of key unrecoverability to the public-key setting. In fact, we consider two distinct versions of key unrecoverability: “Universal Key Unrecoverability” (UKU), where the adversary is tasked to recover the exact challenge private key(s) and “Matching Key Unrecoverability” (MKU), where it suffices to recover suitably equivalent private keys, where we leverage our imperfect correctness notion to define “suitably equivalent”. As one would expect, this relaxed key unrecoverability notion implies the stronger, exact notion up to a small loss related to how we model imperfect correctness (Theorem 1).

For our main notion of multi-instance security, we follow BRT’s identification of left-or-right xor-indistinguishability as the strongest notion and adapt it to the public key setting. As for the symmetric encryption setting studied by BRT, this indistinguishability notion implies the above key unrecoverability notions (Theorem 2); however, the differences between perfect symmetric encryption and imperfect PKE affect the corresponding implications and their proofs.

Finally, we explore an alternative notion, namely real-or-random xor-indistinguishability (ROR). Trivially, left-or-right tightly implies real-or-random and in the multi-instance setting BRT showed that the usual factor-2 loss from the single instance implication between real-or-random to left-or right, becomes an exponential factor- $2^\kappa$  loss. A similar loss is possible in our setting, however, we can also achieve a typically preferable bound of  $\binom{\kappa}{n}2^n$  (Corollary 2).

With suitable notions for multi-instance PKE available, we focus on how to turn a suitably multi-instance secure KEM into a multi-instance secure PKE scheme using hybrid encryption. For key unrecoverability, inheritance is immediate, yet we would like to guarantee good multi-instance indistinguishability (the left-hand branch of Fig. 1). We summarize our findings in Fig. 2.

Our first observation is that we can expand the length of the ephemeral key to any desired length using a pseudorandom extendable output function (XOF). The resulting extendable KEM, or XEM, inherits the multi-instance security of the underlying KEM, provided the XOF is secure against multi-challenge adversaries (Theorem 5). To ensure that the XOF does not become the weakest link, its seed will need to be long enough, which in turn implies that the underlying KEM already needs to output a sufficiently long ephemeral key.

The XOF above of course plays the role of key derivation function, but it is more common that it is modelled as part of any key expansion done by the DEM. Moving it into the KEM allows us to use an information-theoretic DEM, read one-time pad (OTP), irrespective of the message length. The OTP’s properties enable a simplified proof for the security of hybrid encryption (Theorem 6), where the PKE does indeed inherit the multi-instance security of the XEM, with two important caveats. Firstly, the OTP is only passively secure, so the PKE only achieves CPA not CCA security, and secondly, standard KEM indistinguishability only tightly provides real-or-random indistinguishability for the PKE (see the top line of Fig. 2).

Switching to the TagKEM framework [2], or in our case TagXEM, takes care of the first shortcoming and tightly achieves multi-instance ROR-CCA secure

PKE, or IND-CCA non-tightly (Theorem 7). For the PKE to inherit multi-instance IND-CCA security tightly, we introduce a novel KEM indistinguishability notion that more closely matches PKE’s left-or-right idea, namely real-or-permuted (ROP). Finally, we can show tight multi-instance inheritance for the most desirable PKE notion, based on a ROP-secure TagXEM (Theorem 8).

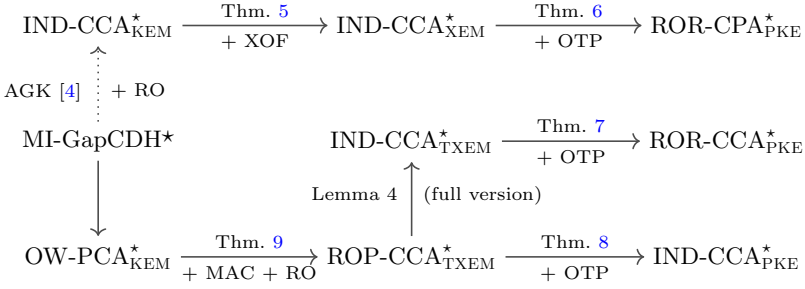
One small hiccup remains, as our KEM-to-XEM result unfortunately only works for classical KEM indistinguishability, not for ROP indistinguishability, nor does it look feasible to convert a KEM or XEM to a TagKEM or TagXEM, respectively, inheriting multi-instance security using standard reductions. Here, the random oracle, as used by AGK to prove their construction secure, comes to the rescue, although rather than looking at Hashed ElGamal under the MI-GapCDH assumption, we consider more general KEMs that are multi-instance one-way under plaintext checking attacks (unfortunately, also at this point we need to restrict to perfect correctness), which we combine with Abe et al.’s TagKEM construction from a KEM and a MAC (message authentication code).

Recalling that the original random oracle [14] was in fact a XOF, we can bake the extendability into the random oracle, including the key needed for an information-theoretic secure MAC. Moreover, the power of the ROM allows proving the stronger ROP indistinguishability just as easily as classical KEM indistinguishability. All in all, with Theorem 9 we achieve a suitably multi-instance secure TagXEM based on a KEM that can be instantiated by Hashed ElGamal. In that case, the security relies on the MI-GapCDH\* assumption, i.e. with corruptions. As an added benefit of using the random oracle, the resulting multi-instance bounds no longer rely on sufficiently long XOF inputs, thus for determining a suitable group size (when instantiating by Hashed ElGamal) the MI-GapCDH\* advantage is leading.

For low granularity, which corresponds to a setting where every user generates its own group as part of its public key, AGK’s technique can easily be extended to include corruptions and in the generic group model we arrive at the same bound for the hardness of MI-GapCDH\*, so with corruptions, as AGK did without corruptions. Unfortunately, for the more realistic high granularity setting, where users share the same (standardized) group, AGK’s proof strategy does not easily allow incorporating corruptions. We provide details in the full version [17].

Thus, we can conclude that XOF-based Hashed ElGamal combined with a suitable information-theoretically secure MAC and the one-time-pad, provides good multi-instance security in the programmable random oracle model and generic group model, provided that users each select their own independent group. We briefly touch upon a concrete interpretation in the full version, where we also informally address AGK’s scaling factor.

**Related Work.** Farshim and Tessaro [20] recently followed up BRT’s line of work on the multi-instance security of password hashing by combining it with the related preprocessing setting. AGK [4] motivated their investigation into multi-instance security by the threat of mass surveillance. The latter had previously motivated Bellare et al. [11] to consider subversion, namely the ease with



**Fig. 2.** An overview of our constructions achieving various flavours of multi-instance security. The left upwards arrow is dotted, as AGK did not consider corruptions.

which a “big brother” might subvert an encryption algorithm by replacing it surreptitiously with a trapdoored one with otherwise identical behaviour.

The multi-instance setting is closely related to the multi-user setting, in which the adversary is tasked with breaking only one rather than  $n$  out of  $\kappa$  possible instances. Multi-user security was introduced by Bellare et al. [7] in the public-key setting, with the goal of deriving concrete security parameters in a more realistic setting. There have been many recent follow-up works, including how the hybrid paradigm generalizes to the setting without corruptions [23], and later with corruptions [33], as well as the construction of tightly-secure authenticated key exchange (AKE) from multi-user KEMs [29]. Various versions of the multi-user GapCDH problem with corruptions were recently proposed and analysed in that context [30].

One definitional subtlety of multi-user security is the number of challenge bits: either a single one, as originally conceived, or many, as typical for the multi-instance setting. The various definitions do not appear to imply each other tightly [26], which slightly hinders regarding the multi-user setting as a special case of the multi-instance setting (due to potential tightness losses).

## 2 Preliminaries

### 2.1 Notation

For a positive integer  $n$ , we write  $[n]$  for the set  $\{1, \dots, n\}$ . We use code-based experiments, where  $\leftarrow$  denotes deterministic assignment and  $\leftarrow \$$  denotes probabilistic assignment. By convention, all sets and lists are initialized empty. For a set  $X$ , we use the shorthand  $X \leftarrow^{\cup} x$  for the operation  $X \leftarrow X \cup \{x\}$ . If  $X$  is a list, then  $X \leftarrow x$  denotes appending the element  $x$  to  $X$ ; to retrieve the  $i$ th element of the list, we write  $X[i]$  where by convention  $X[i] = \emptyset$  for out-of-bounds  $i$ .

We use  $\Pr[\text{Code} : \text{Event} \mid \text{Condition}]$  to denote the conditional probability of *Event* occurring when *Code* is executed, conditioned on *Condition*. We omit *Code* when it is clear from the context and *Condition* when it is not needed.

For Boolean values, we use  $\{\text{true}, \text{false}\}$  and  $\{0, 1\}$  interchangeably, where by convention 1 corresponds to **true**.

When proving relations between notions and security of constructions, we will often refer to simple fully black box (SFBB) reductions. A reduction is fully black box iff it works for all schemes and adversaries, and only accesses them in a black box manner [6, 38] (we leave the black box dependence implicit in our notation). Moreover, if the reduction only runs its adversary once and without rewinding, then the reduction is simple [34].

Finally, the respective games that the adversary and the reduction are playing often have matching (though not identical) oracles; for instance, both may have access to a decryption oracle or a key corruption oracle. We call a reduction type-preserving with respect to, say, a decryption oracle iff the reduction will make decryption queries iff its black-box adversary makes decryption queries. Type-preservation, without explicit mention of any oracles, is implicitly meant to imply for all meaningfully matching oracles (unless otherwise specified).

Type-preservation of reductions appears folklore and can easily be established by inspection. Intuitively, a type-preserving reduction can be used to show simultaneously that CCA security of some kind implies CCA security of another kind and that CPA security of the same kind implies CPA security of the other kind. In Sect. 3.3 we will encounter several reductions that are only partially type-preserving.

## 2.2 PKE Syntax

A public-key encryption scheme PKE consists of four algorithms: the probabilistic key generation algorithm  $\text{PKE.Kg}$ , which takes as input some system parameter  $\text{pm}$  (see also Remark 1) and outputs a public/private key pair  $(\text{pk}, \text{sk})$ ; the deterministic key validation algorithm  $\text{PKE.Check}$ , which takes as input the system parameters  $\text{pm}$  as well as a purported public/private key pair  $(\text{pk}, \text{sk})$  and returns **true** or **false** (see Remark 2 below), the probabilistic encryption algorithm  $\text{PKE.Enc}$ , which on input a public key  $\text{pk}$  and a message  $m \in \mathcal{M}$  (see Remark 3), outputs a ciphertext  $c$ ; and the deterministic decryption algorithm  $\text{PKE.Dec}$ , which on input of a secret key  $\text{sk}$  and a ciphertext  $c$ , outputs either a message  $m$ , or a special symbol  $\perp$  denoting failure.

*Remark 1.* The system parameters  $\text{pm}$  are implicitly input to  $\text{PKE.Enc}$  and  $\text{PKE.Dec}$  as well; for concreteness, they can for instance be the description of an elliptic curve group with generator for an ECDLP-based system or the dimensions and noise sampling algorithm for an LWE-based system. When one is interested in re-phrasing our results in an asymptotic setting, the parameters  $\text{pm}$  will be generated by a probabilistic, polynomial-time algorithm that only takes the security parameter as input.

*Remark 2.* For various modern cryptosystems, especially schemes targeting post-quantum security or tight multi-user security, the relationship between public and private keys is not one-to-one. For instance, a single public key can have various private keys [23] or a single private key can lead to various public keys [16].

Naively, one could check whether a public key and private key belong together by simply verifying whether encrypting and then decrypting a number of random messages always returns the original messages. With imperfect correctness, such a canonical checking algorithm can produce both false positives and false negatives. Yet, it is usually still possible to check whether a private–public key pair matches more directly, which we model by the key validation algorithm `PKE.Check`. We will define both correctness and key unrecoverability in terms of this key validation algorithm.

*Remark 3.* The message space  $\mathcal{M}$  may depend on the parameters  $\mathbf{pm}$ , but for simplicity we assume it independent of the public key  $\mathbf{pk}$ . Often  $\mathcal{M}$  consists of arbitrary length bitstrings, or at least all bitstrings up to some large length (e.g.  $2^{64}$ ) and messages of the same length are deemed equivalent as they are expected to yield ciphertexts of identical lengths. We will model these equivalences more abstractly by assuming that  $\mathbf{pm}$  implicitly defines a number  $\mathbf{m}$  of equivalence classes, together with an efficient method  $\llbracket \cdot \rrbracket : \mathcal{M} \rightarrow [\mathbf{m}]$  to determine the class (e.g. length) of a message and an efficient algorithm to sample uniformly from a given equivalence class. We write  $\sim$  for the equivalence, so for  $m \in \mathcal{M}$ ,  $m \sim m'$  iff  $\llbracket m \rrbracket = \llbracket m' \rrbracket$ .

**Correctness.** Perfect correctness states that for all parameters  $\mathbf{pm}$ , all key pairs  $(\mathbf{pk}, \mathbf{sk})$  that can be output by `PKE.Kg`( $\mathbf{pm}$ ), and all messages  $m \in \mathcal{M}$ , we always have that `PKE.Dec` <sub>$\mathbf{sk}$</sub> (`PKE.Enc` <sub>$\mathbf{pk}$</sub> ( $m$ )) =  $m$ . Yet modern schemes, especially lattice-based ones, often allow a small decryption error, where occasionally decryption will fail or it will return a wrong message.

Various relaxations of correctness have appeared in the literature in order to argue about such schemes as it turns out that some classical results implicitly or subtly relied on perfect correctness. In order for our work to be meaningful for a large range of both classical and modern schemes, we introduce a stronger version of imperfect correctness based on the key validation algorithm.

**Definition 1 (( $\gamma, \delta$ )-Correctness).** *Let  $\gamma, \delta \in [0, 1]$ . Then a public-key encryption scheme `PKE` is called ( $\gamma, \delta$ )-correct iff for all  $\mathbf{pm}$ ,*

1.  $\Pr[(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{PKE.Kg}(\mathbf{pm}) : \text{PKE.Check}(\mathbf{pm}, \mathbf{pk}, \mathbf{sk}) = \text{false}] \leq \gamma$ ;
2. *for all  $(\mathbf{pk}, \mathbf{sk})$  and all  $m \in \mathcal{M}$ , if  $\text{PKE.Check}(\mathbf{pm}, \mathbf{pk}, \mathbf{sk}) = \text{true}$  then*

$$\Pr[\text{PKE.Dec}_{\mathbf{sk}}(\text{PKE.Enc}_{\mathbf{pk}}(m)) \neq m] \leq \delta .$$

Perfect correctness corresponds to  $(0, 0)$ -correctness and any scheme is trivially both  $(1, 0)$ -correct and  $(0, 1)$ -correct. For good schemes  $\gamma$  and  $\delta$  can simultaneously be chosen small, where typically increasing  $\gamma$  allows for decreasing  $\delta$ . As we will see, both  $\gamma$  and  $\delta$  will appear in various bounds, thus allowing larger  $\gamma$  to enable smaller  $\delta$  (or vice versa) might give preferable bounds.

### 3 Multi-instance Security of Public-Key Encryption

#### 3.1 Two Flavours of Key Recovery

The minimal requirement for public-key encryption schemes is that, given a public key, it should be difficult to recover the private key. Although key unrecoverability is a very weak notion theoretically, its study has two main motivations: firstly, many multi-instance attacks target key recovery, and secondly, conceptually the notion is relatively simple, allowing both an instructive introduction of formalizing multi-instance security and an initial comparison between BRT's perfect symmetric encryption and our imperfect public key encryption.

At first sight, the generalization to the multi-instance setting appears immediate: an adversary tries to recover the respective private keys for a number of public keys. BRT introduced universal key unrecoverability (UKU) as a suitable notion for multi-instance security of symmetric encryption. We provide an analogue for public-key encryption, but there are some crucial changes in the game's mechanics (see also Sect. 3.3).

Let  $0 < n \leq \kappa$  be integer parameters, then the universal key unrecoverability experiment  $\text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-uku-cca}\star}(\mathbb{A})$  for public-key encryption scheme PKE and adversary  $\mathbb{A}$  is described in Fig. 3. It generates  $\kappa$  key pairs and provides the public keys to  $\mathbb{A}$ , who is then tasked with recovering exactly  $n$  of the corresponding private keys.

The adversary has access to both a decryption oracle  $\mathcal{D}$  and a key corruption oracle  $\mathcal{K}$ , giving rise to chosen ciphertext attacks with corruptions (CCA $\star$ ; the  $\star$  denotes corruptions). The decryption oracle  $\mathcal{D}(i, c)$  takes as input an index  $i$  and a ciphertext  $c$ , and returns the output of the decryption algorithm  $\text{PKE.Dec}$  on input  $\text{sk}_i$  and  $c$ . The corruption oracle  $\mathcal{K}(i)$  simply takes as input a key index  $i$ , and returns the corresponding private key  $\text{sk}_i$ . The game notes that the key pair with index  $i$  has been corrupted by adding it to the global set  $K$ .

Eventually,  $\mathbb{A}$  outputs a set of key indices  $I$  and a list  $(\text{sk}_i)_{i \in I}$  of guesses of the private keys corresponding to those indices. In order for  $I$  to be eligible, it needs to have cardinality  $n$  without containing any corrupted key pairs, that is, the sets of guessed keys  $I$  and corrupted keys  $K$  should be disjoint. If  $I$  is eligible and every guessed private key matches the corresponding sampled one, the adversary wins the game. In that case, the game halts with output 1; otherwise, it halts with output 0. The advantage is the probability that the game outputs 1.

**Definition 2.** *Let PKE be a public-key encryption scheme. Then the universal key unrecoverability advantage of an adversary  $\mathbb{A}$  is*

$$\text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-uku-cca}\star}(\mathbb{A}) = \Pr \left[ \text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-uku-cca}\star}(\mathbb{A}) = 1 \right],$$

where the experiment is defined in Fig. 3.

Weaker notions emerge by dropping either or both of the two oracles. Without key corruption, standard CCA security results. Without decryption oracle, chosen plaintext security (CPA $\star$  resp. CPA) emerges. As usual, an encryption oracle is superfluous in the PKE setting.

Experiment $\text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-}(u/m)\text{ku-cca}^*}(\mathbb{A})$	Oracle $\mathcal{D}(i, c)$
$(\text{pk}_1, \text{sk}_1), \dots, (\text{pk}_\kappa, \text{sk}_\kappa) \leftarrow \text{\$ PKE.Kg}$ $(\mathbb{I}, (\hat{\text{sk}}_i)_{i \in \mathbb{I}}) \leftarrow \text{\$ } \mathbb{A}^{\mathcal{D}, \mathcal{K}}(\text{pk}_1, \dots, \text{pk}_\kappa)$ <b>if</b> $ \mathbb{I}  \neq n \vee \mathbb{I} \cap \mathbb{K} \neq \emptyset$ <b>then return 0</b> UKU : <b>return</b> $\bigwedge_{i \in \mathbb{I}} \text{sk}_i = \hat{\text{sk}}_i$ MKU : <b>return</b> $\bigwedge_{i \in \mathbb{I}} \text{PKE.Check}(\text{pk}_i, \hat{\text{sk}}_i)$	$m \leftarrow \text{PKE.Dec}_{\text{sk}_i}(c)$ <b>return</b> $m$ <hr style="width: 50%; margin: 10px auto;"/> Oracle $\mathcal{K}(i)$ $\mathbb{K} \xleftarrow{\text{U}} i$ <b>return</b> $\text{sk}_i$

**Fig. 3.** The key recovery experiments  $\text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-uku-cca}^*}(\mathbb{A})$  and  $\text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-mku-cca}^*}(\mathbb{A})$ ; they only differ in their win condition.

For cryptosystems where a single public key may have many matching private keys (such as Cramer–Shoup [19]), universal key unrecoverability is rather weak. Hence, we consider a second, slightly stronger notion of key recovery, in which the recovered private keys are no longer required to be identical to those sampled in the game. Instead, it suffices that each passes the keypair checking algorithm  $\text{PKE.Check}$ ; here we leverage our correctness definition (Definition 1). We call the resulting notion *matching key unrecoverability* (MKU), whose game is included in Fig. 3. That MKU security indeed implies UKU security is captured by Theorem 1 below, where the error term  $\kappa\gamma$  results from the unique correct keys as output by the key generation not always passing the  $\text{PKE.Check}$  algorithm (see the full version for the proof).

**Theorem 1** (MKU  $\longrightarrow$  UKU). *Let  $0 < n \leq \kappa$  be integer parameters and let  $\text{PKE}$  be a  $(\gamma, \delta)$ -correct encryption scheme. Then, there is a type-preserving SFBB reduction  $\mathbb{B}_{\text{mku}}$ , such that for every adversary  $\mathbb{A}_{\text{uku}}$ ,*

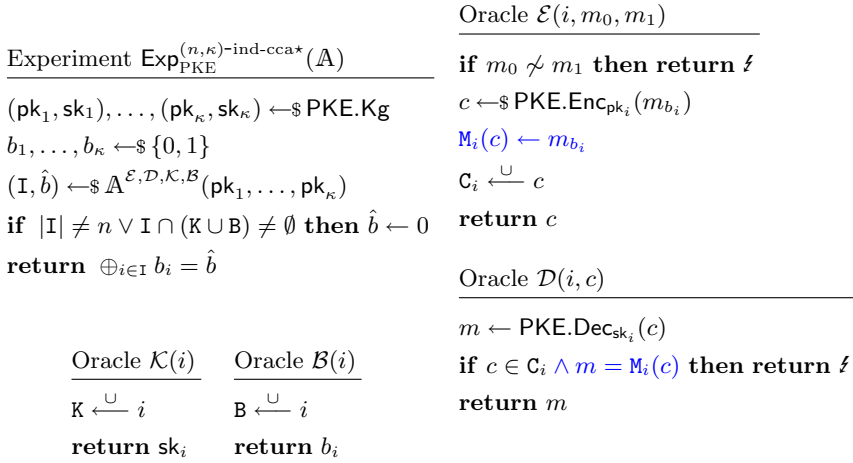
$$\text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-uku-cca}^*}(\mathbb{A}_{\text{uku}}) \leq \text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-mku-cca}^*}(\mathbb{B}_{\text{mku}}) + \kappa\gamma.$$

### 3.2 Left-or-Right XOR Indistinguishability

To capture a stronger notion of security than simply hardness of key recovery, BRT considered various generalizations of indistinguishability to the multi-instance setting. For perfect probabilistic symmetric encryption, they concluded that left-or-right xor-indistinguishability is the strongest notion. Here each key comes with its own challenge bit that determines the left-or-right nature of the corresponding challenge encryption oracle; the adversary is tasked to retrieve the xor of all the challenge bits. In Definition 3, we use our modified game mechanics to adapt left-or-right xor-indistinguishability for potentially non-perfect public-key encryption.

**Definition 3.** *Let  $\text{PKE}$  be a public-key encryption scheme. Then the xor-indistinguishability advantage of an adversary  $\mathbb{A}$  is*

$$\text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A}) = 2 \cdot \Pr \left[ \text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A}) = 1 \right] - 1,$$



**Fig. 4.** Our main notion of multi-instance indistinguishability. In blue the slightly non-standard strengthening of the decryption oracle in case of imperfect correctness. (Colour figure online)

where the experiment is defined in Fig. 4.

In the experiment  $\text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A})$ , the adversary gets access to  $\kappa$  independently drawn public keys and helper oracles  $\mathcal{D}$  and  $\mathcal{K}$  (as described in Sect. 3.1). Furthermore,  $\mathbb{A}$  gets access to a challenge encryption oracle  $\mathcal{E}$  and a separate bit corruption oracle  $\mathcal{B}$ .

On input two equivalent messages  $m_0$  and  $m_1$  and a public key index  $i$ , the challenge encryption oracle returns  $\text{PKE.Enc}_{\text{pk}_i}(m_{b_i})$  where  $b_i$  is the challenge bit associated with the public key indexed by  $i$ . As usual for IND-CCA notions, challenge ciphertexts cannot be queried to the decryption oracle, which we catch on-the-fly [9]. Owing to the imperfect decryption, we allow a slight relaxation: if a challenge ciphertext decrypts incorrectly, we do not suppress the output and essentially allow the query. This relaxation strengthens the notion, but as challenge ciphertexts are honestly generated, the advantage gained by an adversary can be bound by the correctness parameters of the PKE using an identical-until-bad argument; however such a generic approach might not give bounds appropriate for the multi-instance setting.

Eventually, the adversary returns a set  $\mathbb{I}$  of targets and a guess  $\hat{b}$  of the xor of the corresponding challenge bits  $b_i$ . If  $\mathbb{I}$  is a set of  $n$  uncorrupted indices, then intuitively an adversary’s uncertainty about any of the  $n$  challenge bits will be affected in the final guess  $\hat{b}$ , so in that sense  $\hat{b}$  neatly captures an adversary’s need to break  $n$  instances in order to win. If  $\mathbb{I}$  is not a set of  $n$  uncorrupted indices, the game resets  $\mathbb{A}$ ’s guess  $\hat{b}$  to 0, ensuring an adversary gains zero advantage from such a bad  $\mathbb{I}$ .

**The Relationship with Key Recovery.** BRT showed that in their perfect symmetric setting, multi-instance indistinguishability implies multi-instance universal key unrecoverability. While that may sound like a triviality, their proof [13,



App. C] was not entirely straightforward and, to ensure that the advantages carried over neatly, the distinguishing reduction receiving recovered keys needed to amplify its success probability by repeated random challenge encryptions. Their bound ends up with an additive term that corresponds to the likelihood that decrypting using an incorrect key results in the opposite message from the decrypted one.

Our imperfect public key setting is slightly different. On the one hand, the reduction can check the recovered keys with the  $\text{PKE.Check}$  algorithm, yet on the other hand correct keys can still cause incorrect decryptions. As a result, our amplification based on multiple challenge encryptions differs from BRT's, as we move from unanimity to a plurality vote. Furthermore, our reduction can use fixed messages (to match how correctness is defined), which reduces a dependency (in the bound) on the size of the message space. We suspect that our amplification can be tightened further by a combination of exploiting randomness and more fine-tuned voting, coupled with more fine-grained bounding of probabilities.

As is, the complexity of the bound makes its behaviour somewhat opaque and for some parameter choices vacuous (when  $c < 0$ ). The main idea is that  $\mathbb{B}_{\text{ind}}$  can increase  $q$ , the number of challenge encryptions per user, to counteract the losses inferred by large  $n$  and/or large  $\delta$ , with a small penalty to its running time. For  $\delta = 2^{-64}$ ,  $q = 1$  already suffices for  $c > 1/2$  for  $n < 2^{25}$ . In case of perfect correctness for keys that check out, corresponding to  $\delta = 0$ , the bound is completely tight.

**Theorem 2** (IND  $\rightarrow$  MKU). *Let PKE be a  $(\gamma, \delta)$ -correct encryption scheme with  $\delta < 1/2$ . Then there is a type-preserving SFBB reduction  $\mathbb{B}_{\text{ind}}$  such that, for every  $\mathbb{A}_{\text{mku}}$ ,*

$$\text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{B}_{\text{ind}}) \geq c \cdot \text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-mku-cca}^*}(\mathbb{A}_{\text{mku}}),$$

with  $c = 2 \left(1 - 2^q (\delta(1 - \delta))^{\frac{q}{2}}\right)^n - 1$  where  $q \in \mathbb{Z}_{>0}$  is an amplification parameter of the reduction;  $\mathbb{B}_{\text{ind}}$ 's overhead consists of  $q \cdot n$  calls to  $\mathcal{E}$ ,  $n$  offline key checks, and  $q \cdot n$  offline decryptions.

*Proof.* Let  $\mathbb{B}_{\text{ind}}$  run adversary  $\mathbb{A}_{\text{mku}}$  on the same  $\kappa$  public keys as it received itself. Whenever  $\mathbb{A}_{\text{mku}}$  makes a decryption or corruption query,  $\mathbb{B}_{\text{ind}}$  simply forwards the queries to its own oracle, relaying the response back to  $\mathbb{A}_{\text{mku}}$ . Eventually,  $\mathbb{A}_{\text{mku}}$  terminates with output  $(I, (\text{sk}_i)_{i \in I})$  and  $\mathbb{B}_{\text{ind}}$  first confirms whether  $\mathbb{A}_{\text{mku}}$  won, by checking, for all the returned private keys, whether  $\text{PKE.Check}(\text{pk}_i, \hat{\text{sk}}_i)$  holds. If any check fails,  $\mathbb{B}_{\text{ind}}$  halts with output 0.

Let  $m_0$  and  $m_1$  be two distinct yet equivalent messages. Then for all  $i \in I$ ,  $\mathbb{B}_{\text{ind}}$  creates a guess  $\hat{b}_i$  by querying its challenge encryption oracle  $q$  times on those two messages, so  $q$  queries  $\mathcal{E}(i, m_0, m_1)$  resulting in  $c_{ij}$ , for  $j \in [q]$ . It then decrypts those ciphertexts using the private key  $\hat{\text{sk}}_i$  it obtained from  $\mathbb{A}_{\text{mku}}$ , resulting in purported messages  $m_{ij} \leftarrow \text{PKE.Dec}_{\hat{\text{sk}}_i}(c_{ij})$ . If, for a fixed  $i$ , there are strictly more than  $q/2$  appearances of  $m_0$  amongst the  $m_{ij}$ , it sets  $\hat{b}_i$  to 0;

if there are strictly more than  $q/2$  appearances of  $m_1$ , then it sets  $\hat{b}_i$  to 1. If neither message appears more than  $q/2$  times,  $\mathbb{B}_{\text{ind}}$  halts with output 0. Once  $\mathbb{B}_{\text{ind}}$  has created a guess  $\hat{b}_i$  for all  $i \in \mathbb{I}$ , it terminates on output  $(\mathbb{I}, \bigoplus_{i \in \mathbb{I}} \hat{b}_i)$ .

For  $i \in \mathbb{I}$ , let  $\text{Check}_i$  be the event that  $\mathbb{A}_{\text{mku}}$  outputs a key  $\hat{\text{sk}}_i$  that passes the test and let  $\text{Good}_i$  be the event that  $\mathbb{B}_{\text{ind}}$ 's guess  $\hat{b}_i$  actually equals  $b_i$ . Let  $\text{Check}_{\mathbb{I}}$  be the event that all  $\text{Check}_i$  hold (for  $i \in \mathbb{I}$ ) and define  $\text{Good}_{\mathbb{I}}$  analogously.

As  $\mathbb{B}_{\text{ind}}$ 's simulation of  $\text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-mku-cca}^*}$  is perfect, we know that

$$\text{Adv}^{(n,\kappa)\text{-mku-cca}^*}(\mathbb{A}_{\text{mku}}) = \Pr[\text{Check}_{\mathbb{I}}] ,$$

moreover,

$$\begin{aligned} \Pr \left[ \text{Exp}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{B}_{\text{ind}}) = 1 \right] &\geq \Pr[\text{Check}_{\mathbb{I}} \wedge \text{Good}_{\mathbb{I}}] + \Pr[\neg \text{Check}_{\mathbb{I}} \wedge b = 0] \\ &= \Pr[\text{Good}_{\mathbb{I}} \mid \text{Check}_{\mathbb{I}}] \Pr[\text{Check}_{\mathbb{I}}] + \frac{1}{2} (1 - \Pr[\text{Check}_{\mathbb{I}}]) \end{aligned}$$

which implies that

$$\text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{B}_{\text{ind}}) \geq (2 \Pr[\text{Good}_{\mathbb{I}} \mid \text{Check}_{\mathbb{I}}] - 1) \text{Adv}^{(n,\kappa)\text{-mku-cca}^*}(\mathbb{A}_{\text{mku}}) .$$

To bound  $\Pr[\text{Good}_{\mathbb{I}} \mid \text{Check}_{\mathbb{I}}]$  we exploit the correctness definition, specifically that its quantification (Definition 1) ensures that whenever  $\text{Check}_i$  holds, we have that  $\Pr \left[ \text{PKE.Dec}_{\hat{\text{sk}}_i}(\text{PKE.Enc}_{\text{pk}_i}(m)) = m \right] \geq 1 - \delta$ , irrespective of  $m$  and where the probability is only over the randomness of  $\text{PKE.Enc}$ .

If, for a given  $i$ , decryption is correct strictly more than  $q/2$  times, then we are guaranteed that  $\text{Good}_i$  occurs. If we let  $B(q, p)$  be the binomial distribution over  $q$  trials and with probability  $p$ , then

$$\Pr[\text{Good}_i \mid \text{Check}_i] \geq \Pr \left[ B(q, (1 - \delta)) > \frac{q}{2} \right]$$

and, as this bound only relies on the randomness of the challenge encryption oracle, guaranteed independent for differing  $i$ , we may conclude that

$$\Pr[\text{Good}_{\mathbb{I}} \mid \text{Check}_{\mathbb{I}}] \geq \left( \Pr \left[ B(q, (1 - \delta)) > \frac{q}{2} \right] \right)^n .$$

Finally, we note that

$$\Pr \left[ B(q, (1 - \delta)) > \frac{q}{2} \right] \geq 1 - 2^q (\delta(1 - \delta))^{\frac{q}{2}}$$

by a standard application of known bounds on binomial tails, requiring  $\delta \leq 1/2$  (see details below). Plugging in all the various bounds recovers the theorem statement.

For the binomial tail bound, we use the Chernoff–Hoeffding bound [27], which states that, for a binomial distribution  $B(q, p)$  over  $q$  trials and with probability  $p$ , and any  $k$  satisfying  $p < \frac{k}{q} < 1$  the tail bound

$$\Pr \left[ B(q, p) \geq k \right] \leq \exp \left[ -qD \left( \frac{k}{q} \parallel p \right) \right]$$

holds, where  $D(a\|b)$  is the Kullback–Leibler divergence defined as  $D(a\|b) = a \ln \left(\frac{a}{b}\right) + (1-a) \ln \left(\frac{1-a}{1-b}\right)$ .

We further use the trick that  $\Pr[B(q, (1-\delta)) > \frac{q}{2}] = 1 - \Pr[B(q, \delta) \leq \frac{q}{2}]$ , so the relevant Kullback–Leibler divergence becomes

$$\begin{aligned} D\left(\frac{1}{2}\left\|\delta\right.\right) &= \frac{1}{2} \ln\left(\frac{\frac{1}{2}}{\delta}\right) + \left(1 - \frac{1}{2}\right) \ln\left(\frac{(1-\frac{1}{2})}{1-\delta}\right) \\ &= \frac{1}{2} \ln\left(\frac{1}{2\delta}\right) + \frac{1}{2} \ln\left(\frac{1}{2(1-\delta)}\right) \\ &= \ln\left[\left(\frac{1}{4\delta(1-\delta)}\right)^{\frac{1}{2}}\right], \end{aligned}$$

which allows us to compute the bound

$$\begin{aligned} \Pr\left[B(q, (1-\delta)) > \frac{q}{2}\right] &\geq 1 - \exp\left[-qD\left(\frac{1}{2}\left\|\delta\right.\right)\right] \\ &= 1 - \exp\left[-q \ln\left[\left(\frac{1}{4\delta(1-\delta)}\right)^{\frac{1}{2}}\right]\right] \\ &= 1 - 2^q (\delta(1-\delta))^{\frac{q}{2}}. \end{aligned}$$

□

**Corollary 1** (IND  $\longrightarrow$  UKU). *Let PKE be a  $(\gamma, \delta)$ -correct encryption scheme with  $\delta < 1/2$ . Then there is a type-preserving SFBB reduction  $\mathbb{B}_{\text{ind}}$  such that, for every  $\mathbb{A}_{\text{uku}}$ ,*

$$\text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{B}_{\text{ind}}) \geq c \cdot \text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-uku-cca}^*}(\mathbb{A}_{\text{uku}}) - \kappa\gamma,$$

with  $c, q$ , and  $\mathbb{B}_{\text{ind}}$ 's overhead as above (Theorem 2).

### 3.3 Alternative Mechanisms

As we mentioned before, our mechanism to capture multi-instance security differs slightly from those used by BRT and AGK, respectively, even when accounting for changes in primitive and correctness. At first sight, the differences might appear mostly cosmetic, though there are some subtleties involved.

**The BRT Notion: Requiring  $n = \kappa$ , Possibly Corrupted, Targets.** BRT require an adversary to return the xor of all bits, but allow those bits or corresponding users to be corrupted. Figure 5 reflects the small change needed in the code of our security experiment to match BRT's mechanism (ignoring a minor, inconsequential difference, as BRT have a single, merged corruption oracle that returns both key and bit). As motivation for including corruptions, BRT discuss the scenario that, say, half of the keys generated are hopelessly insecure: an adversary breaks the insecure half and corrupts the rest, thus being successful.

Experiment $\text{Exp}_{\text{PKE}}^{(\leq \kappa, \kappa)\text{-ind-cca}^*}(\mathbb{A})$	Experiment $\text{Exp}_{\text{PKE}}^{(\geq n, \kappa)\text{-ind-cca}^*}(\mathbb{A})$
4 : <b>if</b> $ \mathbf{I}  \neq \kappa$ <b>then</b> $\hat{b} \leftarrow 0$	4 : <b>if</b> $ \mathbf{I}  < n \vee \mathbf{I} \cap (\mathbf{K} \cup \mathbf{B}) \neq \emptyset$ <b>then</b> $\hat{b} \leftarrow 0$

**Fig. 5.** The main differences between our mechanism for multi-instance indistinguishability (Fig. 4) and prior art revolve around line 4: BRT’s experiment  $\text{Exp}_{\text{PKE}}^{(\leq \kappa, \kappa)\text{-ind-cca}^*}(\mathbb{A})$  (left) and AGK’s experiment  $\text{Exp}_{\text{PKE}}^{(\geq n, \kappa)\text{-ind-cca}^*}(\mathbb{A})$  (right). The differences are highlighted in blue. (Colour figure online)

Moreover, they mention that their choice implies security under a corruptionless notion with dynamically chosen  $\mathbf{I}$ .

Although the implication is of course true, and something can be said to target the strongest possible notion, corruptions have a habit of creating complications for reductions and provable security in general. Yet, we believe the inclusion of corruptions, or not, should reflect the threat model of the adversary and that choice should be orthogonal to the number of users being targeted. BRT, instead of having an explicit hardness parameter  $n$ , restrict an adversary to make at most  $q_c$  corruption queries to avoid trivial wins when  $q_c = \kappa$ . Yet, whether the resulting, intuitive hardness will or should then match  $n = \kappa - q_c$ , is unclear.

We address the equivalence between BRT’s mechanism and our general mechanism (with corruptions) in Lemmas 1 and 2. Both lemmas have in common that the respective reductions may make up to  $\kappa - n$  additional bit corruptions. In other words, the reductions are not type-preserving, making the equivalence somewhat sloppy. As an aside, using techniques similar to those to prove Theorem 2, the key corruption oracle could be used (at a loss) to simulate the bit corruption oracle instead (see the full version for the proofs).

**Lemma 1 (main notion  $\implies$  BRT).** *Let  $n \leq \kappa$  and  $q_c \leq \kappa - n$ . Then there is an SFBB reduction  $\mathbb{B}$  such that, for every adversary  $\mathbb{A}$  making at most  $q_c$  corruption oracle calls,*

$$\text{Adv}_{\text{PKE}}^{(\leq \kappa, \kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{B}),$$

where  $\mathbb{B}$  makes at most  $\kappa - n$  additional bit corruption oracle calls.

**Lemma 2 (BRT  $\implies$  main notion).** *Let  $n \leq \kappa$ . Then there is an SFBB reduction  $\mathbb{B}$  such that, for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{PKE}}^{(\leq \kappa, \kappa)\text{-ind-cca}^*}(\mathbb{B}),$$

where  $\mathbb{B}$  makes at most  $\kappa - n$  additional bit corruption oracle calls.

**The AGK Notion: Allowing More than  $n$  Targets without Corruptions.** When AGK studied KEMs in the multi-instance setting, they used a xor notion with the  $n$  as the *minimum* number of targets to attack (out of  $\kappa$  possible) as an explicit parameter; moreover, an adversary would not have access to any

corruption oracles. Figure 5 reflects the small change needed in the code of our security experiment to match AGK’s mechanism with corruptions added (where we fixed a minor bug in their code; rather than setting  $\hat{b} \leftarrow 0$  their experiment would immediately return 0 instead, inadvertently granting an adversary that deliberately returns a compromised handle the significant advantage of  $-1$ ).

Absent corruptions, AGK indicated that for some pathological schemes, breaking more targets might paradoxically be easier than breaking fewer [3, App. C]. In those cases, the freedom to return a set  $I$  of cardinality greater than  $n$  would make life easier for an adversary, leading to a stronger notion.

In the presence of corruptions, requiring the adversary to target exactly  $n$  users as we do is without loss of generality. As an example, if an adversary can figure out the xor of  $n + 1$  honest bits, it can bit-corrupt any single one of these  $n + 1$ , and xor the resulting bit out of the initial guess to obtain a final one on  $n$  bits instead. We formalize this intuition below.

**Lemma 3 (main notion  $\implies$  AGK $\star$ ).** *There is an SFBB adversary  $\mathbb{B}$  such that, for every  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}}^{(\geq n, \kappa)\text{-ind-cca}\star}(\mathbb{A}) \leq \text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-ind-cca}\star}(\mathbb{B}).$$

*If  $\mathbb{A}$  returns a list of  $n'$  targets,  $\mathbb{B}$  makes  $n' - n$  additional calls to its bit corruption oracle.*

### 3.4 Real-or-Random XOR Indistinguishability

An alternative notion of indistinguishability, known as real-or-random indistinguishability (ROR), sees the adversary tasked with figuring out whether a challenge ciphertext contains the adversarially chosen message  $m$  or an unknown, randomly chosen message. The game  $\text{Exp}_{\text{PKE}}^{(n, \kappa)\text{-ror-cca}\star}$  is exactly as in Fig. 4, apart from the challenge encryption oracle  $\mathcal{E}_{\text{ROR}}(i, m)$ , which sets  $m_0 \leftarrow m$  and  $m_1 \leftarrow \$[m]$  to then call (left-or-right)  $\mathcal{E}(i, m_0, m_1)$ .

By construction, left-or-right indistinguishability easily implies real-or-random indistinguishability. That statement is as true in the multi-instance setting as it is in the classical single-user setting. Conversely, in the single-user setting, it has long been established that the reduction from ROR to IND loses a factor 2 [8]. However, BRT showed that in the multi-instance setting, the factor 2 blows up exponentially to, in their case,  $2^\kappa$ . Yet, BRT argue that this exponential loss is not as bad as it might seem, given that the multi-instance advantages are supposed to be exponentially smaller than their single-user counterparts. Thus, reductions incurring losses exponential in  $\kappa$  or  $n$  can still be valuable.

To adapt BRT’s reduction to our setting, we require  $n = \kappa$ , implying that  $\mathbb{A}$  cannot access its corruption oracles. Otherwise, corruptions would make the reduction noticeable once at least one  $b_i$  is set to 1, potentially influencing an adversary’s behaviour in unpredictable ways (see the full version for the proof).

**Theorem 3.** *There is an SFBB reduction  $\mathbb{B}$  such that, for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}}^{(\kappa,\kappa)\text{-ind-cca}}(\mathbb{A}) \leq 2^\kappa \cdot \text{Adv}_{\text{PKE}}^{(\kappa,\kappa)\text{-ror-cca}}(\mathbb{B}),$$

where  $\mathbb{B}$  additionally draws  $\kappa$  bits uniformly at random.

Furthermore, a reduction playing an  $(n, n)$  game can exploit an adversary playing a  $(n, \kappa)$  game by guessing in advance the set  $I$  of targets that the adversary will return. A correct guess allows the reduction to simulate the remaining keys without being noticed (see the full version for the proof).

**Theorem 4.** *There is an SFBB reduction  $\mathbb{B}$  such that, for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot \text{Adv}_{\text{PKE}}^{(n,n)\text{-ind-cca}}(\mathbb{B}).$$

$\mathbb{B}$ 's overhead consists of generating  $\kappa - n$  fresh keypairs, sampling  $\kappa - n$  bits, and choosing a subset of  $[\kappa]$  of cardinality  $n$  uniformly at random.

Composing Theorem 3 and 4, we obtain the following bound.

**Corollary 2.** (ROR  $\implies$  IND). *There is an SFBB reduction  $\mathbb{B}$  such that, for any adversary  $\mathbb{B}$ ,*

$$\text{Adv}_{\text{PKE}}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot 2^n \cdot \text{Adv}_{\text{PKE}}^{(n,n)\text{-ror-cca}}(\mathbb{B}).$$

$\mathbb{B}$ 's overhead consists of generating  $\kappa - n$  fresh keypairs, sampling  $\kappa$  bits, and choosing a subset of  $[\kappa]$  of cardinality  $n$  uniformly at random.

An alternative bound losing a factor  $2^\kappa$  is possible by combining Theorem 3 with Lemma 2, however a simple analysis shows that whenever  $n < \kappa/5$  the corollary above is preferable.

At first glance, an exponential-looking loss of  $2^\kappa$  might seem severe, potentially rendering the resulting bound vacuous. Yet, as BRT already highlighted, the multi-instance advantages themselves might vanish exponentially in  $n$ , making the bounds relevant for the notions being compared. Nonetheless, tighter bounds still matter; unfortunately achieving even tighter bounds in the general case seems challenging [5, 12].

## 4 Inheriting Multi-instance Security

### 4.1 TagKEM: Definition and Notion of Security

Our goal is to turn the AGK multi-instance secure KEM into a PKE. Yet, for the construction of hybrid encryption, the more general TagKEMs, where encapsulation is split into two algorithms (TKEM.Key and TKEM.Enc) have proven more powerful [2]: intuitively speaking, splitting the algorithm allows the tag and consequently the key encapsulation to depend on the data encapsulation, making

Experiment $\text{Exp}_{\text{TXEM}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A})$	Oracle $\mathcal{C}(i, \ell)$
$(\text{pk}_1, \text{sk}_1), \dots, (\text{pk}_\kappa, \text{sk}_\kappa) \leftarrow \$ \text{TXEM.Kg}$ $b_1, \dots, b_\kappa \leftarrow \$ \{0, 1\}$ $(\mathbb{I}, \hat{b}) \leftarrow \$ \mathbb{A}^{\mathcal{C}, \mathcal{E}, \mathcal{D}, \mathcal{K}, \mathcal{B}}(\text{pk}_1, \dots, \text{pk}_\kappa)$ <b>if</b> $ \mathbb{I}  \neq n \vee \mathbb{I} \cap (\mathbb{K} \cup \mathbb{B}) \neq \emptyset$ <b>then</b> $\hat{b} \leftarrow 0$ <b>return</b> $\oplus_{i \in \mathbb{I}} b_i = \hat{b}$	$(K_0, \sigma) \leftarrow \$ \text{TXEM.Key}_{\text{pk}_i}(\ell)$ $\mathbf{E}_i \leftarrow \widehat{\langle \sigma, K_0 \rangle}$ $K_1 \leftarrow \$ \{0, 1\}^\ell$ <b>return</b> $K_{b_i}$
Oracle $\mathcal{D}(i, \langle c, \tau \rangle, \ell)$	Oracle $\mathcal{E}(i, j, \tau)$
$K \leftarrow \text{TXEM.Dec}_{\text{sk}_i}(c, \tau, \ell)$ <b>if</b> $\text{P}_i(c, \tau) \neq \emptyset$ $K' \leftarrow \text{P}_i(c, \tau), \ell' \leftarrow \min\{\ell,  K' \}$ <b>else</b> $K' \leftarrow \varepsilon, \ell' \leftarrow 0$ <b>if</b> $\langle c, \tau \rangle \in \mathbf{C}_i \wedge K[\ell'] = K'[\ell']$ <b>return</b> $\perp$ <b>return</b> $K$	<b>if</b> $\mathbf{E}_i[j] = \emptyset$ <b>then return</b> $\perp$ $\langle \sigma, K \rangle \leftarrow \mathbf{E}_i[j], \mathbf{E}_i[j] \leftarrow \emptyset$ $c \leftarrow \$ \text{TXEM.Enc}(\sigma, \tau)$ $\text{P}_i(c, \tau) \leftarrow K$ $\mathbf{C}_i \leftarrow \bigcup \langle c, \tau \rangle$ <b>return</b> $c$
	Oracle $\mathcal{K}(i)$ Oracle $\mathcal{B}(i)$
	$\mathbb{K} \leftarrow \bigcup i$ $\mathbb{B} \leftarrow \bigcup i$ <b>return</b> $\text{sk}_i$ <b>return</b> $b_i$

**Fig. 6.** Multi-instance indistinguishability notion for TXEM. In blue the same strengthening as in Fig. 4 in the case of imperfect correctness, with a slightly more complex admin to accommodate tags and length extension. We take  $K[\ell]$  to mean the first  $\ell$  bits of  $K$  and  $\varepsilon$  as the empty string. (Colour figure online)

CCA security of the hybrid construction easier to achieve (cf. the Kurosawa-Desmedt scheme [31]). In Definition 4 we introduce a further generalization, called TagXEM, by allowing extendable output lengths for the ephemeral keys produced by the TagXEM.

**Definition 4 (TagXEM).** A *TagXEM* is a tuple of algorithms (TXEM.Kg, TXEM.Key, TXEM.Enc, TXEM.Dec, TXEM.Check), where long-term key generation TXEM.Kg on input pm outputs a random keypair (pk, sk); ephemeral key generation TXEM.Key on input pk and  $\ell \in \mathbb{Z}_{>0}$ , outputs a random ephemeral key  $K \in \{0, 1\}^\ell$  and an internal state  $\sigma$ , subsequently encapsulation TXEM.Enc on input a state  $\sigma$  and a tag  $\tau \in \mathcal{T}$ , deterministically outputs an encapsulation  $c$ , or a special symbol  $\perp$  denoting failure. The deterministic decapsulation algorithm TXEM.Dec takes input a private key sk, an encapsulation  $c$ , a tag  $\tau$ , and a length  $\ell$ , and outputs either a key  $K \in \{0, 1\}^\ell$  or  $\perp$  to denote failure. Finally, the deterministic TXEM.Check takes as input the system parameters pm as well as a purported public/private key pair (pk, sk) and returns true or false.

If we restrict to a single value  $\ell$ , the usual notion of TagKEMs appears; moreover if we restrict to a single value of  $\tau$ , the TXEM.Key and TXEM.Enc algorithms can

be merged into a single key encapsulation mechanism, leading to normal KEMs (or XEMs if the variable output length is still incorporated). Consequently, the correctness and security definitions for the more general TagXEMs, as discussed throughout this section, imply corresponding definitions for KEM, XEM, and TagKEM.

For correctness, we allow the effective tag space  $\mathcal{T}_\ell$  to depend on the length  $\ell$  of the ephemeral key. Similarly to Definition 1, we define  $(\gamma, \delta)$ -correctness for TagXEM. To ensure correctness for all  $\tau$ , including those that depend on  $K$ ,  $\tau$ 's quantifier sits inside the probability statement.

**Definition 5 (( $\gamma, \delta$ )-Correctness TagXEM).** *Let  $\gamma, \delta \in [0, 1]$ . Then a tag extendable-output key encapsulation mechanism TXEM is called  $(\gamma, \delta)$ -correct iff*

1.  $\Pr[(pk, sk) \leftarrow \$TXEM.Kg(pm) : TXEM.Check(pm, pk, sk) = \text{false}] \leq \gamma;$
2. *if  $TXEM.Check(pm, pk, sk) = \text{true}$  then for all  $\ell \in \mathbb{Z}_{>0}$  it holds that*

$$\Pr \left[ (K, \sigma) \leftarrow \$TXEM.Key_{pk}(\ell) : \exists \tau \in \mathcal{T}_\ell \text{ s.t. } \begin{array}{l} c \leftarrow TXEM.Enc(\sigma, \tau) \\ TXEM.Dec_{sk}(c, \tau, \ell) \neq K \end{array} \right] \leq \delta .$$

For security, Abe et al.'s notion of TagKEM indistinguishability [2] transfers easily to the multi-instance setting. The relevant game is given in Fig. 6, where we also made the necessary changes to deal with the variable output length of TagXEMs, plus the strengthening of  $\mathcal{D}$  in the case of imperfect correctness (cf. Sect. 3.2).

**Definition 6.** *Let TXEM be a TagXEM. Then the xor-indistinguishability advantage of an adversary  $\mathbb{A}$  is*

$$\text{Adv}_{TXEM}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A}) = 2 \cdot \Pr \left[ \text{Exp}_{TXEM}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A}) = 1 \right] - 1 ,$$

where the experiment is defined in Fig. 6.

If we fix  $\ell$  and set  $\mathcal{T}_\ell$  to a single element, the notion captures multi-instance security for standard KEMs, which is near equivalent (see Sect. 3.3) the notion that AGK used. In other words, provided MI-gapCDH is hard, their construction achieves  $(n, \kappa)$ -IND-CCA security in the random oracle  $\kappa$  model, but only for fixed  $\ell$  and trivial  $\mathcal{T}_\ell$  [4, Thm. 2].

## 4.2 Extending the Output of a TagKEM

First, we show how combining a TagKEM with a fixed output length and a suitable pseudorandom extendable output function (XOF), yields a TagXEM that inherits the MI security of the underlying KEM. Recall that a XOF, for instance SHAKE128 and SHAKE256 as standardized by NIST [35], is a function  $F : \mathcal{X} \times \mathbb{Z}_{>0} \rightarrow \{0, 1\}^*$  for some finite domain  $\mathcal{X}$  that on input a seed  $s \in \mathcal{X}$  and a desired output length  $\ell$ , outputs a value  $y \in \{0, 1\}^\ell$ . Moreover, if  $\ell < \ell'$ , then  $F(s, \ell)$  is a prefix of  $F(s, \ell')$  for all  $s$ . This prefix preservation is not a requirement



TXEM.Key <sub>pk</sub> (ℓ)	TXEM.Enc(σ', τ)	TXEM.Dec(c, τ, ℓ)
$(K^{\text{kem}}, \sigma) \leftarrow \text{\$TKEM.Key}_{\text{pk}}$ $K^{\text{xem}} \leftarrow F(K^{\text{kem}}, \ell)$ $\sigma' \leftarrow \langle \sigma, \ell \rangle$ <b>return</b> $(K^{\text{xem}}, \sigma')$	$\langle \sigma, \ell \rangle \leftarrow \sigma'$ <b>if</b> $\tau \notin \mathcal{T}_\ell$ : <b>return</b> $\perp$ $c \leftarrow \text{TKEM.Enc}(\sigma, \tau)$ <b>return</b> $c$	<b>if</b> $\tau \notin \mathcal{T}_\ell$ : <b>return</b> $\perp_{\text{TAG}}$ $K^{\text{kem}} \leftarrow \text{TKEM.Dec}(c, \tau)$ <b>if</b> $K^{\text{kem}} = \perp$ : <b>return</b> $\perp_{\text{KEM}}$ $K^{\text{xem}} \leftarrow F(K^{\text{kem}}, \ell)$ <b>return</b> $K^{\text{xem}}$

**Fig. 7.** A TagXEM TXEM from a TagKEM TKEM with keyspace  $\{0, 1\}^k$  and a XOF with seed space  $\mathcal{X} = \{0, 1\}^k$ . The key generation algorithm TXEM.Kg is unchanged from TKEM.Kg.

of our constructions; rather we model the property to ensure SHAKE128 and SHAKE256 are suitable real-world instantiations.

As security notion for a XOF  $F$  we use its multi-challenge pseudorandomness, which is a standard distinguishing advantage  $\text{Adv}_F^{\text{psrnd}}(\mathbb{A})$ : an adversary needs to distinguish between either a real oracle that, on input a desired length  $\ell$ , samples a seed  $s \leftarrow \text{\$X}$  uniformly at random and returns  $F(s, \ell)$ , or an ideal oracle that, on input said  $\ell$ , simply returns a uniformly sampled string of length  $\ell$ .

The construction of the TagXEM is given in Fig. 7 and the security claim follows in Theorem 5 (see the full version for the proof). If the PsRND advantage of  $F$  is sufficiently small, then TXEM inherits the multi-instance security of TKEM; moreover, as the result holds for arbitrary  $\mathcal{T}$  and  $\mathcal{T}_\ell$ , it holds for the trivial spaces, yielding a slightly simpler XEM from KEM result.

**Theorem 5.** *Let TKEM be a  $(\gamma, \delta)$ -correct TagKEM sampling keys from  $\{0, 1\}^k$  and with tagspace  $\mathcal{T}$ , let  $F : \{0, 1\}^k \times \mathbb{Z}_{>0} \rightarrow \{0, 1\}^*$  be a XOF, and let TXEM be a TagXEM as given in Fig. 7 for arbitrary  $\mathcal{T}_\ell \subseteq \mathcal{T}$ . Then TXEM is  $(\gamma, \delta)$ -correct, and there are SFBB reductions  $\mathbb{B}$  and  $\mathbb{C}$  such that, for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{TXEM}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{TKEM}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{B}) + 2 \cdot \text{Adv}_F^{\text{psrnd}}(\mathbb{C}).$$

If  $\mathbb{A}$  calls  $\mathbb{C}$   $q_c$  times and  $\mathbb{D}$   $q_d$  times, then  $\mathbb{B}$ 's overhead consists of at most  $q_c + q_d$  evaluations of  $F$ , while  $\mathbb{C}$ 's overhead consists of doing  $\kappa$  executions of TKEM.Kg, at most  $q_c$  executions of TKEM.Key and TKEM.Enc, and at most  $q_d$  executions of TKEM.Dec.

One concern is whether the PsRND advantage of  $F$  will be sufficiently small. Suppose  $k$  is the output length of the underlying TagKEM. A generic attacker would always be able to fix  $\ell > k$  and evaluate  $F$  for, say,  $N$  seeds offline in the hope of colliding with any of the challenge evaluations. The PsRND distinguishing advantage of such an adversary is of order  $(q_c + q_d)N/2^k$ , indicating that the underlying TagKEM already needs to provide keys long enough for Theorem 5 to yield meaningful multi-instance security.

$\text{PKE.Enc}_{\text{pk}}(m)$	$\text{PKE.Dec}_{\text{sk}}(\langle c_1, c_2 \rangle)$
$(K, c_1) \leftarrow \$\text{XEM.Enc}_{\text{pk}}( m )$ $c_2 \leftarrow K \oplus m$ <b>return</b> $\langle c_1, c_2 \rangle$	$K \leftarrow \text{XEM.Dec}_{\text{sk}}(c_1,  c_2 )$ <b>if</b> $K = \perp$ <b>then return</b> $\perp$ $m \leftarrow K \oplus c_2$ <b>return</b> $m$
$\text{PKE}'.\text{Enc}_{\text{pk}}(m)$	$\text{PKE}'.\text{Dec}_{\text{sk}}(\langle c_1, c_2 \rangle)$
$(K, \sigma) \leftarrow \$\text{TXEM.Key}_{\text{pk}}( m )$ $c_2 \leftarrow K \oplus m$ $c_1 \leftarrow \text{TXEM.Enc}(\sigma, c_2)$ <b>return</b> $\langle c_1, c_2 \rangle$	$K \leftarrow \text{TXEM.Dec}_{\text{sk}}(c_1, c_2,  c_2 )$ <b>if</b> $K = \perp$ <b>then return</b> $\perp$ $m \leftarrow K \oplus c_2$ <b>return</b> $m$

**Fig. 8.** Two hybrid encryption schemes: PKE (top row) is a conventional hybrid scheme combining a XEM with the OTP to yield a CPA-secure PKE, while PKE' (bottom row) combines a TagXEM with the OTP to yield a CCA-secure PKE. The key generation and checking algorithms are equivalent to their XEM resp. TXEM counterparts.

### 4.3 A PKE Inheriting (Tag)XEM Security

As a multi-instance secure XEM provides us with ephemeral keys of any desired length, we can combine it with an information-theoretic DEM in order to achieve PKE. Here we opt for the one-time-pad (OTP), as it is the simplest and best-known primitive providing perfect secrecy. The beauty of the OTP is that whether you switch out the ephemeral key for a uniform random one, or the message for a uniform random one, the resulting ciphertext distribution is the same. It allows the PKE to tightly inherit the MI-security of the XEM, albeit yielding only real-or-random security under chosen-plaintext attacks. The construction is provided in full in Fig. 8 (top row); the security claim is captured in Theorem 6 (see the full version for the proof).

**Theorem 6 (ROR-CPA PKE).** *Let XEM be a  $(\gamma, \delta)$ -correct XEM, and let PKE be a hybrid encryption scheme as given in Fig. 8. Then PKE is  $(\gamma, \delta)$ -correct, and there is a type-preserving SFBB reduction  $\mathbb{B}$  such that for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}}^{(n, \kappa)\text{-ror-cpa}^*}(\mathbb{A}) \leq \text{Adv}_{\text{XEM}}^{(n, \kappa)\text{-ind-cpa}^*}(\mathbb{B}).$$

One might hope that adding information-theoretic MACs to the DEM would result in the inheritance of CCA security, but that is easier said than shown. For instance, the usual proof technique of a game hop where all decryption queries are disallowed does not work: after breaking only a single KEM private key,

---

Oracle  $\mathcal{C}_{\text{ROP}}(i, \ell, \Pi)$

$(K_0, \sigma) \leftarrow \$ \text{TXEM.Key}_{\text{pk}_i}(\ell)$

$E_i \leftarrow \sigma$

$K_1 \leftarrow \Pi(K_0)$

**return**  $K_{b_i}$

**Fig. 9.** Fig. 6 is upgraded to  $\text{Exp}_{\text{TXEM}}^{(n, \kappa)\text{-rop-cca}^*}$  by letting  $\mathcal{C}_{\text{ROP}}$  replace  $\mathcal{C}$ .

the reduction will be found out as not being faithful. Sadly, a single-instance break (of the reduction) suffices to show that reduction cannot demonstrate multi-instance security.

Luckily, TagKEMs allow for a modified hybrid scheme for which the DEM no longer needs to satisfy CCA security for the resulting PKE to be guaranteed CCA-secure: in the single-instance setting, if the TagKEM is CCA-secure, then so is the PKE [2]. We upgrade the construction to use TagXEMs and the OTP in Fig. 8 (bottom row) and show its multi-instance inheritance in Theorem 7 (see the full version for the proof).

**Theorem 7 (ROR-CCA PKE).** *Let TXEM be a  $(\gamma, \delta)$ -correct TagXEM, and let PKE' be a hybrid encryption scheme as given in Fig. 8. Then PKE' is  $(\gamma, \delta)$ -correct, and there is a type-preserving SFBB reduction  $\mathbb{B}$  such that for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}'}^{(n, \kappa)\text{-ror-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{TXEM}}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{B}).$$

While encouraging, the claim that the constructed PKE inherits the multi-instance security of the TagXEM is dampened by the exponential separation between the ROR security notion and IND, as argued in Sect. 3.4. Indeed, extrapolating to the latter notion by combining Theorem 7 with Corollary 2, we have only achieved the following bound.

**Corollary 3.** *Let TXEM be a  $(\gamma, \delta)$ -correct TagXEM, and let PKE' be a hybrid encryption scheme as given in Fig. 8. Then PKE' is  $(\gamma, \delta)$ -correct, and there is a type-preserving SFBB reduction  $\mathbb{B}$  such that for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}'}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot 2^n \cdot \text{Adv}_{\text{TXEM}}^{(n, n)\text{-ind-cca}}(\mathbb{B}),$$

where  $\mathbb{B}$ 's overhead is dominated by generating  $\kappa - n$  fresh keypairs, sampling  $\kappa$  bits, and choosing a subset of  $[\kappa]$  of cardinality  $n$  uniformly at random.

#### 4.4 Real-or-Permuted: A Strengthened Notion for KEM Security

If we want to achieve an IND-CCA PKE more tightly, we seem to need a different notion of security for our TagXEMs. What could such a notion look like?

Our solution is a novel, stronger KEM notion, which we will refer to as “real-or-permuted”, or ROP for short. Figure 9 provides the crucial new challenge oracle. The adversary has to guess whether a tentative  $K$  is the one encapsulated under  $c$ , or whether an adaptively chosen permutation has been applied to it. As permutations preserve the distribution of the sampling space, there are no choices of  $\Pi$  that make the game generically and trivially winnable.

Technically, we need to specify how the adversary provides  $\Pi$  such that it is guaranteed, or can be checked, to be a permutation. Hence, formally we define ROP with respect to a class of permutations  $\mathcal{P}$ , reminiscent of for instance key-dependent message [24] or related-key attack [10] definitions. We require that membership  $\Pi \in \mathcal{P}$  is easy to check (e.g. ROP can simply index an element in  $\mathcal{P}$ ) and that, by definition,  $\mathcal{P}$  can be verified to indeed only contain permutations. For our main results, it suffices if  $\mathcal{P}$  is the class of one-time pads, in the sense that  $\Pi$  specifies the key (or pad) of the one-time pad enciphering. Henceforth, we will assume that ROP is defined with respect to that class, unless explicitly stated otherwise.

The new notion ROP and IND relate to each other much the same way as IND and ROR for PKE. It is not hard to see that ROP tightly implies IND, whereas the other direction seems to incur the same loss as the ROR-to-IND implication for PKE (see the full version). For completeness, ROP lends itself equally well to XEMs and KEMs, or notions without corruptions or a decryption oracle. Finally, if any of the above primitives are constructed using an IND-secure PKE (e.g. using a Fujisaki–Okamoto style transform [21, 22, 28]), then achieving ROP is as easy as achieving IND: simply let  $K$  be the “left” message, and  $\Pi(K)$  be the “right”!

### 4.5 PKE’ Tightly Inherits IND-CCA Security

Using ROP in place of IND, we are able to show directly that the PKE constructions of Fig. 8 are IND-CPA resp. IND-CCA secure, by (as before) giving a (Tag)XEM reduction that provides a perfect simulation for the PKE adversary.

The crucial observation is that for any pair of messages  $m_0, m_1 \in \{0, 1\}^\ell$ , there exist a permutation  $\Pi_{m_0 \rightarrow m_1}$  on  $\{0, 1\}^\ell$  such that the message encapsulations are related as  $K \oplus m_1 = \Pi_{m_0 \rightarrow m_1}(K) \oplus m_0$ . Namely, the permutation that on input  $K$ , outputs  $m_0 \oplus m_1 \oplus K$  (see the full version for the proof).

**Theorem 8 (IND-CCA PKE).** *Let TXEM be a  $(\gamma, \delta)$ -correct TagXEM, and let PKE’ be a hybrid encryption scheme as given in Fig. 8. Then PKE’ is  $(\gamma, \delta)$ -correct, and there is a type-preserving SFBB reduction  $\mathbb{B}$  such that for every adversary  $\mathbb{A}$ ,*

$$\text{Adv}_{\text{PKE}'}^{(n, \kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{TXEM}}^{(n, \kappa)\text{-rop-cca}^*}(\mathbb{B}).$$

We leave it to the reader to verify that as before, employing a ROP-CPA XEM in place of the TagXEM yields IND-CPA security for the PKE of Fig. 8 (top row), by adapting the proof of Theorem 6 to the above. We again stress that using an information-theoretically CCA-secure DEM together with a CCA XEM does not seem to yield a proof of CCA inheritance to the PKE (see Sect. 4.3).

<hr style="border: 0.5px solid black; margin-bottom: 5px;"/> <b>TXEM.Kg</b> $(pk', sk') \leftarrow \$ \text{KEM.Kg}$ $pk \leftarrow pk'$ $sk \leftarrow \langle pk', sk' \rangle$ <b>return</b> $(pk, sk)$	<hr style="border: 0.5px solid black; margin-bottom: 5px;"/> <b>TXEM.Key<sub>pk</sub>(<math>\ell</math>)</b> $(K^{\text{kem}}, c) \leftarrow \$ \text{KEM.Enc}_{pk}$ $\ell' \leftarrow \ell + \ell_{\text{mackey}}$ $K^{\text{mac}} \  K^{\text{xem}} \leftarrow F(pk, c, K^{\text{kem}}, \ell')$ $\sigma \leftarrow \langle c, K^{\text{mac}} \rangle$ <b>return</b> $K^{\text{xem}}$
<hr style="border: 0.5px solid black; margin-bottom: 5px;"/> <b>TXEM.Check(pk, sk)</b> $\langle pk', sk' \rangle \leftarrow sk$ <b>if</b> $pk \neq pk'$ <b>then return</b> 0 <b>return</b> $\text{KEM.Check}(pk', sk')$	<hr style="border: 0.5px solid black; margin-bottom: 5px;"/> <b>TXEM.Dec<sub>sk</sub>(<math>\langle c, \text{mac} \rangle, \tau, \ell</math>)</b> $\langle pk', sk' \rangle \leftarrow sk$ $K^{\text{kem}} \leftarrow \text{KEM.Dec}_{sk'}(c)$ <b>if</b> $K^{\text{kem}} = \perp$ <b>then return</b> $\perp$ $\ell' \leftarrow \ell + \ell_{\text{mackey}}$ $K^{\text{mac}} \  K^{\text{xem}} \leftarrow F(pk', c, K^{\text{kem}}, \ell')$ <b>if</b> $\text{MAC}_{K^{\text{mac}}}(\tau) \neq \text{mac}$ <b>then return</b> $\perp$ <b>return</b> $K^{\text{xem}}$
<hr style="border: 0.5px solid black; margin-bottom: 5px;"/> <b>TXEM.Enc(<math>\sigma, \tau</math>)</b> $\langle c, K^{\text{mac}} \rangle \leftarrow \sigma$ $\text{mac} \leftarrow \text{MAC}_{K^{\text{mac}}}(\tau)$ <b>return</b> $\langle c, \text{mac} \rangle$	

**Fig. 10.** A TagXEM from a KEM, a MAC, and an XOF  $F$ .

#### 4.6 TagXEM from a KEM, a MAC, and a Random Oracle

With Theorem 8, we achieved what we set out to do: demonstrating tight MI inheritance from a TagXEM to an IND-CCA PKE. However, AGK only showed how to construct an IND-CCA KEM, providing a reduction to the MI-GapCDH assumption in the programmable random oracle model. Without the crucial support of tags, our construction only achieves CPA security. Furthermore, Theorem 5 does *not* easily transfer to the ROP setting: it is not clear how to combine a ROP-CCA KEM with a XOF to yield a ROP-CCA XEM.

We complete the picture by providing a TagXEM construction from a KEM, a MAC, and a XOF. Our construction (Fig. 10) is inspired by Abe et al.'s TagKEM construction [2] and we show that with an information-theoretic MAC, if the KEM is perfectly correct, has unique encapsulations [25] and is multi-instance one-way secure under plaintext-checking attacks (OW-PCA), then the TagXEM is ROP-CCA secure in the programmable random oracle model (to model the XOF). Before stating our concrete security result (Theorem 9), we will define the relevant concepts and advantages below.

**Preliminaries.** One-wayness for KEMs tasks an adversary to retrieve the ephemeral key that has been encapsulated, given the public key and the encapsulation. In the multi-instance setting, an adversary has access to many public keys and various encapsulations per public key and endeavours to find ephemeral keys for encapsulations for as many different public keys as possible (no reward for breaking multiple encapsulations under the same public key).

$\text{Exp}_{\text{KEM}}^{(n,\kappa)\text{-ow-pca}^*}(\mathbb{A})$	$\mathcal{E}(i)$
$(\text{pk}_1, \text{sk}_1), \dots, (\text{pk}_\kappa, \text{sk}_\kappa) \leftarrow \$ \text{KEM.Kg}$	$(K, c) \leftarrow \$ \text{KEM.Enc}_{\text{pk}_i}$
$(\mathbb{I}, (j_i, \hat{K}_i)_{i \in \mathbb{I}}) \leftarrow \$ \mathbb{A}^{\mathcal{E}, \mathcal{P}, \mathcal{K}}(\text{pk}_1, \dots, \text{pk}_\kappa)$	$\text{P}_i \xleftarrow{\text{c}} K$
<b>if</b> $ \mathbb{I}  \neq n \vee \mathbb{I} \cap \mathbb{K} \neq \emptyset$ <b>then return</b> 0	<b>return</b> $c$
<b>return</b> $\bigwedge_{i \in \mathbb{I}} \text{P}_i[j_i] = \hat{K}_i$	$\mathcal{K}(i)$
$\mathcal{P}(i, c, K)$	$\mathbb{K} \xleftarrow{\cup} i$
$K' \leftarrow \text{KEM.Dec}_{\text{sk}_i}(c)$	<b>return</b> $\text{sk}_i$
<b>return</b> $K = K'$	

**Fig. 11.** Multi-instance one-way security in the presence of plaintext checking attacks.

Plaintext-checking attacks (PCA) were introduced by Okamoto and Pointcheval [36, Definition 8] in a single-user public key encryption setting. Intuitively, PCA provides the adversary access to an oracle that, on input a pair  $(m, c)$  determines whether  $c$  encrypts  $m$  or not; more formally [1], the oracle checks whether  $c$  decrypts to  $m$  or not. In the context of KEMs, the PCA oracle takes a pair  $(K^{\text{kem}}, c)$  as input and determines whether  $c$  decapsulates to  $K^{\text{kem}}$  or not. The multi-user or multi-instance generalization is straightforward and the definition (in its modern decryption incarnation) inherently deals with imperfect correctness in the decryption.

Definition 7 considers one-wayness under plaintext checking attacks. For standard ElGamal KEM, where a (multiplicative) discrete-log group with generator  $g$  and of prime order  $q$  is given as part of the parameters, a public key consists of  $h = g^x$  with  $x \leftarrow \$ \mathbb{Z}_q$  the private key, and an encapsulation outputs  $(K^{\text{kem}}, c) = (h^r, g^r)$  for random  $r \leftarrow \$ \mathbb{Z}_q$ , the one-wayness problem (in the single-user case) is equivalent to the computational Diffie–Hellman (CDH) problem. The plaintext checking oracle allows an adversary to learn, for group elements  $(k, c)$  of its choice, whether  $k = c^x$  or not. The corresponding hardness assumption for OW-PCA is known as the Strong CDH assumption. An even stronger assumption is the GapCDH assumption, where an adversary instead can use an oracle that determines whether a quadruple of group elements is a Diffie–Hellman tuple or not.

**Definition 7 (OW-PCA).** *Let KEM be a key encapsulation mechanism. Then the one-way advantage under plaintext-checking attacks of an adversary  $\mathbb{A}$  is*

$$\text{Adv}_{\text{KEM}}^{(n,\kappa)\text{-ow-pca}^*}(\mathbb{A}) = \Pr \left[ \text{Exp}_{\text{KEM}}^{(n,\kappa)\text{-ow-pca}^*}(\mathbb{A}) = 1 \right],$$

where the experiment is defined in Fig. 11.

In addition to perfect correctness and OW-PCA security, the security reduction for our construction (Theorem 9) relies on two further properties of the underlying KEM. Unique encapsulation captures that for a fixed public key and ephemeral key, the encapsulation corresponding to that ephemeral key is unique

(without saying anything about how to compute it). Unique encapsulations have been used before, for instance by Heuer et al. [25] (see also Remark 4 below).

**Definition 8 (Unique Encapsulation).** *Let KEM be a perfectly correct KEM. Then it has unique encapsulations iff*

$$\Pr \left[ \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{\$ KEM.Kg} \\ (K_0^{kem}, c_0) \leftarrow \text{\$ KEM.Enc}_{\text{pk}} : K_0^{kem} = K_1^{kem} \wedge c_0 \neq c_1 \\ (K_1^{kem}, c_1) \leftarrow \text{\$ KEM.Enc}_{\text{pk}} \end{array} \right] = 0 .$$

The second additional property we require from the KEM is that collisions amongst encapsulations (under a single randomly drawn public key) are suitably rare. Definition 9 captures the relevant probability of a  $k$ -way encapsulation collision. If a KEM is perfectly correct with unique encapsulations, then colliding encapsulations are equivalent to colliding ephemeral keys; if, as is usually the case, these ephemeral keys are furthermore chosen uniformly at random from a finite set  $\mathcal{X}$ , we can upper bound  $\epsilon_k(q)$  by  $q^k/|\mathcal{X}|^{k-1}$  using a standard bound on  $k$ -way collisions (see e.g. [37, Appendix B]).

**Definition 9 (Encapsulation Multi-Collisions).** *Let KEM be a KEM, and let  $q, k \in \mathbb{Z}_{>1}$  be parameters. Then the  $k$ -out-of- $q$  encapsulation multi-collision probability is*

$$\epsilon_k(q) = \Pr \left[ \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{\$ KEM.Kg} \\ \forall_{i \in [q]} (K_i^{kem}, c_i) \leftarrow \text{\$ KEM.Enc}_{\text{pk}} : \exists_{J \subseteq [q], |J|=k} \forall_{i, j \in J} c_i = c_j \end{array} \right] .$$

For completeness, we also present definitions of a deterministic message authentication code, so we dispense with an explicit verification algorithm in Definition 10 (for concreteness, we restrict to bitstrings for both keys and tags, of length  $\ell_{\text{mkey}}$  and  $\ell_{\text{mac}}$  respectively), and an information-theoretic notion of forgeries (Definition 11) where we use the same parameter  $k$  as above (or rather  $k - 1$  in Theorem 9), but this time to denote the number of valid message–tag pairs available to an adversary. The usual choice is  $k = 1$ , e.g. when considering strongly universal<sub>2</sub> hash functions, but Wegman and Carter [40] already investigated  $k > 1$ . Provided  $\ell_{\text{mkey}}$  is large enough (at least  $k \cdot \ell_{\text{mac}}$ ), one can achieve  $\hat{\epsilon}_k = 2^{-\ell_{\text{mac}}}$ , which is optimal.

**Definition 10 (Message Authentication Code (MAC)).** *A message authentication code MAC is a pair of algorithms MAC.Kg and MAC.Mac, where MAC.Kg randomly generates a  $K^{\text{mac}} \in \{0, 1\}^{\ell_{\text{mkey}}}$ , and the deterministic MAC.Mac takes a key  $K^{\text{mac}}$  and a message  $m \in \mathcal{M}$  to output tag  $\text{mac} \leftarrow \text{MAC.Mac}_{K^{\text{mac}}}(m) \in \{0, 1\}^{\ell_{\text{mac}}}$ .*

**Definition 11 (Information-Theoretic MAC Forgeries).** *Let MAC be given and let  $k \in \mathbb{Z}_{\geq 0}$  be a parameter, then the forging advantage after observing  $k$  valid message–tag pairs is defined as*

$$\hat{\epsilon}_k = \max_{\substack{\forall_i \in \{0\} \cup [k] \\ (m_i, \text{mac}_i)}} \Pr \left[ \text{MAC.Mac}_{K^{\text{mac}}}(m_0) = \text{mac}_0 \mid \forall_{i \in [k]} \text{MAC.Mac}_{K^{\text{mac}}}(m_i) = \text{mac}_i \right] .$$

**Security Claim.** With all elements in place, we can state the security of Fig. 10’s TXEM, in Theorem 9 (see the full version for the proof). The security bound depends on a tuning parameter  $k$  that feeds into both the collision probability of the underlying KEM and the forgery advantage of the MAC, with opposite effects. The ability to tune the bound therefore allows some flexibility when instantiating the three underlying primitives KEM, MAC, and XOF: for fixed  $q_c$ , increasing  $k$  will result in a smaller upper bound on  $\epsilon_k(q_c)$ , but to ensure that  $\hat{\epsilon}_{k-1}$  does not dominate, it might then be necessary to increase the key size  $\ell_{\text{mkey}}$  (and possibly tag size  $\ell_{\text{mac}}$ ) of the information-theoretic MAC (see Corollary 5 for a concrete instantiation). Otherwise, instantiating the information-theoretic MAC and the XOF is relatively straightforward (with the usual ROM caveats for the latter).

**Theorem 9.** *Let TXEM be as in Fig. 10, let KEM be a perfectly correct KEM with unique encapsulations, and let  $k \in \mathbb{Z}_{>1}$ . Then there is an SFBB reduction  $\mathbb{B}$  such that, for all  $\mathbb{A}$  that makes  $q_c$  challenge and  $q_d$  decryption oracle queries,*

$$\text{Adv}_{\text{TXEM}}^{(n,\kappa)\text{-rop-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{KEM}}^{(n,\kappa)\text{-ow-pca}^*}(\mathbb{B}) + 2(q_d \hat{\epsilon}_{k-1} + \epsilon_k(q_c))$$

*in the programmable random oracle model, where  $\hat{\epsilon}_{k-1}$  is the forging advantage after observing  $k - 1$  valid message–tag pairs (Definition 11) and  $\epsilon_k(q_c)$  is the  $k$ -out-of- $q_c$  encapsulation multi-collision probability of KEM (Definition 9). If  $\mathbb{A}$  makes  $q_f$  queries to the random oracle, then  $\mathbb{B}$  makes at most  $q_f$  queries to its plaintext checking oracle.*

The proof borrows some ideas already used to prove AGK’s Theorem 2. In fact, it is relatively straightforward to recast AGK’s Theorem 2 as the multi-instance version of a OW-PCA KEM plus a programmable random oracle yielding an IND-CCA KEM, although the presence of the error terms  $\hat{\epsilon}_{k-1}$  and especially  $\epsilon_k(q_c)$  render recovery of AGK’s Theorem 2 as a special case of our Theorem 9 not immediate.

Combining Theorem 8 and 9 in Corollary 4, we can finally conclude that our construction yields a PKE inheriting the multi-instance security of the underlying KEM (for parameter regimes where the loss term does not dominate).

**Corollary 4.** *Let PKE’ be as in Fig. 8, let the underlying TagXEM be as in Fig 10, let KEM be a perfectly correct KEM with unique encapsulations, and let  $k \in \mathbb{Z}_{>1}$ . Then, there is an SFBB reduction  $\mathbb{B}$  such that, for all  $\mathbb{A}$  that makes  $q_c$  challenge and  $q_d$  decryption oracle queries,*

$$\text{Adv}_{\text{PKE}' }^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \text{Adv}_{\text{KEM}}^{(n,\kappa)\text{-ow-pca}^*}(\mathbb{B}) + 2(q_d \hat{\epsilon}_{k-1} + \epsilon_k(q_c))$$

*in the programmable random oracle model, where  $\hat{\epsilon}_{k-1}$  is the forging advantage after observing  $k - 1$  valid message–tag pairs (Definition 11) and  $\epsilon_k(q_c)$  is the  $k$ -out-of- $q_c$  encapsulation multi-collision probability of KEM (Definition 9). If  $\mathbb{A}$  makes  $q_f$  queries to the random oracle, then  $\mathbb{B}$  makes at most  $q_f$  queries to its plaintext checking oracle.*



*Remark 4.* The resulting construction is remarkably similar to the PKE studied by Heuer et al. [25] in the context of selective opening attacks (and to a lesser extent its predecessor by Steinfeld et al. [39] and successor by Lai et al. [32]). They too use a random oracle to derive a MAC key and a one-time pad from an ephemeral KEM key. The only two differences are that Heuer et al. do not consider arbitrary length messages and that their random oracle outputs  $K^{\text{xem}}\|K^{\text{mac}}$ , i.e. the opposite order from what we do.

For fixed length messages, the order in which those two keys are output does not matter. However, when moving to arbitrary length-messages, the order of the XOF output does matter. Outputting  $K^{\text{xem}}\|K^{\text{mac}}$  instead would allow a length extension attack enabling the adversary to recover the MAC key, at which point producing forgeries would be trivial.

In a way, the construction is quite brittle that these small details matter. Another example of brittleness is that our reduction for Theorem 9 requires  $\perp$  produced from a KEM decryption error to be indistinguishable from a failed MAC verification. In implementations, a timing attack might well break this requirement.

*Remark 5.* The proof of Theorem 9 does rely on perfect correctness of the underlying KEM, thus excluding many popular post-quantum KEMs based on the hardness of LWE. Having said that, establishing the post-quantum security of TXEM would require a proof in the quantum random oracle model [15]. We leave the construction of a post-quantum TagXEM as an enticing open problem.

**A Concrete Instantiation.** We conclude by providing a concrete bound for the construction when instantiating with low granularity ElGamal KEM on groups of size  $\geq p$ . ElGamal KEM satisfies perfect correctness and unique encapsulation (ensuring compatibility with Theorem 9) and produces uniformly random group elements as ephemeral keys, so  $\epsilon_k(q_c) \leq q^k/p^{k-1}$ . Furthermore, the relevant multi-instance OW-PCA security can be linked to the low granularity MI-GapCDH problem with corruptions (Theorem 12 of the full version). By extending AGK’s low granularity bound [4, Thm. 6] to include corruptions (Thm. 11 of the full version) and combining with Corollary 4, we arrive at a clean information-theoretic bound (Corollary 5) in the generic group and programmable random oracle model. To keep the bound easier to interpret, we assume that the adversary makes at most  $\sqrt{p}$  queries to the encryption and decryption oracles; realistically, an adversary will be able to make far more offline queries  $q$  to its generic group and for  $q \approx \sqrt{p}$  a single discrete logarithm instance can already be broken. In a similar vein, the requirement that each group instance receive at least  $\max\{60 \log_2 p, \sqrt{q}r/2\}$  group operation calls (allowing some simplifications in the MI-GapCDH bound) is a reasonable one, as already argued by AGK, given that the number of group operations performed by an ElGamal adversary is “typically large”.

**Corollary 5.** *Let PKE’ be as in Fig. 8, let the underlying TagXEM be as in Fig 10, let KEM be instantiated as low granularity ElGamal (see the full version*

for details) and let  $p$  be a lower bound on the generated groups. Let  $k \in \mathbb{Z}_{>1}$ , let  $\text{MAC}$  be an information-theoretic MAC with key length  $\ell_{\text{mackey}}$  and output length  $\ell_{\text{mac}}$  and satisfying  $\hat{\epsilon}_{k-1} = 2^{-\ell_{\text{mac}}}$ . Then, for any information-theoretic  $\mathbb{A}$  that makes at most  $\sqrt{p}$  challenge oracle queries, at most  $\sqrt{p}$  decryption oracle queries,  $q_f$  queries to the random oracle, and a total of  $q$  queries to the group-operation oracles with at least  $\max\{60 \log_2 p, \sqrt{qf}/2\}$  queries per group instance, it holds that

$$\text{Adv}_{\text{PKE}'}^{(n,\kappa)\text{-ind-cca}^*}(\mathbb{A}) \leq \left( \frac{4 \cdot e \cdot q^2}{n^2 \cdot p} \right)^n + 2 \left( \frac{\sqrt{p}}{2^{\ell_{\text{mac}}}} + \frac{1}{p^{\frac{k}{2}-1}} \right)$$

in the programmable random oracle and generic group model.

For the construction to exhibit meaningful multi-instance security, we want the upper bound on the adversary's advantage to diminish with increasing  $n$ . Since the second term on the right hand side of Corollary 5 is independent of  $n$ , the first term has to dominate for advantages of interest. Thus, for a fixed  $p$ , we want to set  $\ell_{\text{mac}}$  and  $k$  so that, irrespective of  $n$ , we do not really care about the other two terms, where  $\ell_{\text{mac}}$  directly corresponds to the PKE's ciphertext expansion and increasing  $k$  will require longer ephemeral keys as output by the XOF to ensure that  $\ell_{\text{mackey}} \geq k \cdot \ell_{\text{mac}}$ . To minimize overhead, having both terms equal is optimal, corresponding to  $2\ell_{\text{mac}} = (k-1) \log_2 p$ . Some reasonable options are then  $(\ell_{\text{mac}}, k) = (\log_2 p, 3)$  or  $(\ell_{\text{mac}}, k) = (3/2 \log_2 p, 4)$ .

Alternatively, the bound can be interpreted in terms of the scaling factor, which focuses on the minimum resources needed to achieve an overwhelming advantage (see the full version for details). In that case, the second term, being independent of  $n$ , is manifestly of little interest for either of our suggested parameter choices.

## References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 332–352. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46447-2\\_15](https://doi.org/10.1007/978-3-662-46447-2_15)
2. Abe, M., Gennaro, R., Kurosawa, K.: Tag-KEM/DEM: a new framework for hybrid encryption. *J. Cryptol.* **21**(1), 97–130 (2007). <https://doi.org/10.1007/s00145-007-9010-x>
3. Auerbach, B., Giacon, F., Kiltz, E.: Everybody's a target: scalability in public-key encryption. *Cryptology ePrint Archive, Report 2019/364* (2019). <https://eprint.iacr.org/2019/364>
4. Auerbach, B., Giacon, F., Kiltz, E.: Everybody's a target: scalability in public-key encryption. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 475–506. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45727-3\\_16](https://doi.org/10.1007/978-3-030-45727-3_16)
5. Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49896-5\\_10](https://doi.org/10.1007/978-3-662-49896-5_10)

6. Baecher, P., Brzuska, C., Fischlin, M.: Notions of black-box reductions, revisited. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 296–315. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-42033-7\\_16](https://doi.org/10.1007/978-3-642-42033-7_16)
7. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45539-6\\_18](https://doi.org/10.1007/3-540-45539-6_18)
8. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055718>
9. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: when and how should challenge decryption be disallowed? *J. Cryptol.* **28**(1), 29–48 (2013). <https://doi.org/10.1007/s00145-013-9167-4>
10. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-39200-9\\_31](https://doi.org/10.1007/3-540-39200-9_31)
11. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_1](https://doi.org/10.1007/978-3-662-44371-2_1)
12. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 312–329. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32009-5\\_19](https://doi.org/10.1007/978-3-642-32009-5_19)
13. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. *Cryptology ePrint Archive*, Report 2012/196 (2012). <https://eprint.iacr.org/2012/196>
14. 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 93, pp. 62–73. ACM Press, November 1993. <https://doi.org/10.1145/168588.168596>
15. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-25385-0\\_3](https://doi.org/10.1007/978-3-642-25385-0_3)
16. Bos, J., et al.: Crystals - Kyber: a CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS P), pp. 353–367 (2018). <https://doi.org/10.1109/EuroSP.2018.00032>
17. Brunetta, C., Heum, H., Stam, M.: Multi-instance secure public-key encryption. *Cryptology ePrint Archive*, Report 2022/909 (2022). <https://eprint.iacr.org/2022/909>
18. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**(1), 167–226 (2003)
19. 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>

20. Farshim, P., Tessaro, S.: Password hashing and preprocessing. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 64–91. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77886-6\\_3](https://doi.org/10.1007/978-3-030-77886-6_3)
21. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-48405-1\\_34](https://doi.org/10.1007/3-540-48405-1_34)
22. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptol.* **26**(1), 80–101 (2011). <https://doi.org/10.1007/s00145-011-9114-1>
23. Giaccon, F., Kiltz, E., Poettering, B.: Hybrid encryption in a multi-user setting, revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10769, pp. 159–189. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-76578-5\\_6](https://doi.org/10.1007/978-3-319-76578-5_6)
24. Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007, pp. 466–475. ACM Press, October 2007. <https://doi.org/10.1145/1315245.1315303>
25. Heuer, F., Jager, T., Kiltz, E., Schäge, S.: On the selective opening security of practical public-key encryption schemes. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 27–51. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46447-2\\_2](https://doi.org/10.1007/978-3-662-46447-2_2)
26. Heum, H., Stam, M.: Tightness subtleties for multi-user PKE notions. In: Paterson, M.B. (ed.) IMACC 2021. LNCS, vol. 13129, pp. 75–104. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92641-0\\_5](https://doi.org/10.1007/978-3-030-92641-0_5)
27. Hoeffding, W.: Probability inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.* **58**, 13–30 (1963)
28. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 341–371. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70500-2\\_12](https://doi.org/10.1007/978-3-319-70500-2_12)
29. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 117–146. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77870-5\\_5](https://doi.org/10.1007/978-3-030-77870-5_5)
30. Kiltz, E., Pan, J., Riepel, D., Ringerud, M.: Multi-user CDH problems and the concrete security of NAXOS and HMQV. In: Rosulek, M. (ed.) CT-RSA 2023. Springer, Heidelberg (2023, to appear). <https://eprint.iacr.org/2023/115>
31. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_26](https://doi.org/10.1007/978-3-540-28628-8_26)
32. Lai, J., Yang, R., Huang, Z., Weng, J.: Simulation-based bi-selective opening security for public key encryption. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13091, pp. 456–482. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92075-3\\_16](https://doi.org/10.1007/978-3-030-92075-3_16)
33. Lee, Y., Lee, D.H., Park, J.H.: Tightly CCA-secure encryption scheme in a multi-user setting with corruptions. *Des. Codes Cryptogr.* **88**(11), 2433–2452 (2020)
34. Lewko, A., Waters, B.: Why proving HIBE systems secure is difficult. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 58–76. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-55220-5\\_4](https://doi.org/10.1007/978-3-642-55220-5_4)
35. NIST: SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication 202, NIST, August 2015

36. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–174. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45353-9\\_13](https://doi.org/10.1007/3-540-45353-9_13)
37. Preneel, B.: Analysis and design of cryptographic hash functions. Ph.D. thesis, KU Leuven, February 1993
38. 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](https://doi.org/10.1007/978-3-540-24638-1_1)
39. Steinfeld, R., Baek, J., Zheng, Y.: On the necessity of strong assumptions for the security of a class of asymmetric encryption schemes. In: Batten, L., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 241–256. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45450-0\\_20](https://doi.org/10.1007/3-540-45450-0_20)
40. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.* **22**, 265–279 (1981)



# Unidirectional Updatable Encryption and Proxy Re-encryption from DDH

Peihan Miao<sup>1</sup>, Sikhar Patranabis<sup>2</sup>(✉), and Gaven Watson<sup>3</sup>

<sup>1</sup> Brown University, Providence, USA  
peihan\_miao@brown.edu

<sup>2</sup> IBM Research India, Bangalore, India  
sikhar.patranabis@ibm.com, sikharpatranabis@gmail.com

<sup>3</sup> Meta, Menlo Park, USA  
gavenwatson@meta.com

**Abstract.** Updatable Encryption (UE) and Proxy Re-encryption (PRE) allow *re-encrypting* a ciphertext from one key to another in the symmetric-key and public-key settings, respectively, without decryption. A longstanding open question has been the following: do *unidirectional* UE and PRE schemes (where ciphertext re-encryption is permitted in only one direction) necessarily require stronger/more structured assumptions as compared to their bidirectional counterparts? Known constructions of UE and PRE seem to exemplify this “gap” – while bidirectional schemes can be realized as relatively simple extensions of public-key encryption from standard assumptions such as DDH or LWE, unidirectional schemes typically rely on stronger assumptions such as FHE or indistinguishability obfuscation (iO), or highly structured cryptographic tools such as bilinear maps or lattice trapdoors.

In this paper, we bridge this gap by showing the first feasibility results for realizing unidirectional UE and PRE from a new generic primitive that we call Key and Plaintext Homomorphic Encryption (KPHE) – a public-key encryption scheme that supports additive homomorphisms on its plaintext and key spaces simultaneously. We show that KPHE can be instantiated from DDH. This yields the first constructions of unidirectional UE and PRE from DDH.

Our constructions achieve the strongest notions of *post-compromise security* in the standard model. Our UE schemes also achieve “backwards-leak directionality” of key updates (a notion we discuss is equivalent, from a security perspective, to that of unidirectionality with no-key updates). Our results establish (somewhat surprisingly) that unidirectional UE and PRE schemes satisfying such strong security notions *do not*, in fact, require stronger/more structured cryptographic assumptions as compared to bidirectional schemes.

## 1 Introduction

Cryptographic encryption is a powerful tool for ensuring data confidentiality. A common security guarantee offered by any encryption scheme (either symmetric-key or public-key) is the following: encrypted data can only be decrypted using a

certain secret key. However, a limitation of traditional encryption schemes is that once data is encrypted, it is generally hard to allow a third party to transform the ciphertext so that it can be decrypted with a different key, without sharing either the original or the new secret key with the third party.

Re-encryption schemes such as Proxy Re-encryption (PRE) [BBS98] and Updatable Encryption (UE) [BLMR13] circumvent this limitation by enabling a public transformation of ciphertexts from encryption under one key to that of another, while protecting the underlying secret keys. Classic applications for such schemes include key rotation for secure outsourced storage [BBB+12, Pay18], access control, the delegation of email access, and many more.

**Proxy Re-encryption (PRE).** PRE is a public-key encryption scheme which enables a party Alice, with the help of a proxy, to re-encrypt her ciphertexts for decryption by an alternate party Bob. To facilitate re-encryption, Alice and Bob, with key pairs  $(pk_A, sk_A)$  and  $(pk_B, sk_B)$  respectively, will together compute a re-encryption key  $rk_{AB}$  and then provide this to the proxy. Whenever the proxy needs to perform re-encryption, it can use  $rk_{AB}$  to transform a ciphertext encrypted under  $pk_A$  into a ciphertext encrypted under  $pk_B$ . Security of the PRE scheme guarantees that the proxy learns nothing about the underlying plaintext during the re-encryption process.

**Updatable Encryption (UE).** UE was introduced by Boneh et al. [BLMR13] to address the problem of key rotation for secure outsourced storage. UE addresses re-encryption by using similar techniques to those of PRE, with two main differences: (1) UE is a symmetric-key encryption scheme, and (2) UE typically only allows sequential updates. More specifically, in UE we divide time into a series of epochs. In the first epoch a fresh symmetric key  $k_0$  is chosen and used to encrypt all data. When we rotate a key from  $k_{e-1}$  to  $k_e$ , we transition to the next epoch by calculating an update token  $\Delta_e$ . All new ciphertexts are encrypted under the new key  $k_e$  and all existing ciphertexts  $ct_{e-1}$  are re-encrypted using the update token  $\Delta_e$  so that they can be decrypted by  $k_e$ . The benefit of this approach is that the storage server can perform the re-encryption of data using the update token without the risk of exposing any plaintext data.

There are two variants of UE schemes, *ciphertext-dependent* schemes [EPRS17, BEKS20] and *ciphertext-independent* schemes [LT18, KLR19, BDGJ20, Jia20]. In ciphertext-dependent UE schemes, the update token  $\Delta_{e, ct_{e-1}}$  depends on the ciphertext  $ct_{e-1}$  to be updated, while in ciphertext-independent schemes, the update token  $\Delta_e$  is generated independent of the updated ciphertext, hence a single token can be used to update all ciphertexts on the storage server. In the rest of the paper, when we refer to UE, we mean a ciphertext-independent scheme unless otherwise specified.

**Directionality of PRE and UE.** Re-encryption schemes are either *bidirectional* or *unidirectional*. A scheme is said to be bidirectional if a re-encryption key/update token can be used to re-encrypt a ciphertext to either the next party/epoch or the previous party/epoch. In contrast, the re-encryption key/update token of a unidirectional scheme can only be used to re-encrypt a ciphertext to the next party/epoch and *not* the previous. So far we have only discussed the

directionality within the context of ciphertext updates. The (uni)directionality with regards to keys differs slightly between PRE and UE, as we discuss next.

**Bidirectionality vs. Unidirectionality in PRE.** In bidirectional PRE schemes, the re-encryption key  $rk_{AB}$  from Alice to Bob is generated from Alice’s key-pair  $(pk_A, sk_A)$  and Bob’s key-pair  $(pk_B, sk_B)$ . Given  $rk_{AB}$  along with  $sk_A$  (resp.  $sk_B$ ), it is usually possible to derive  $sk_B$  (resp.  $sk_A$ ). In unidirectional PRE schemes, the re-encryption key  $rk_{AB}$  is derived from  $((pk_A, sk_A), pk_B)$ ; Bob’s secret key  $sk_B$  is not used. In fact, given the re-encryption key  $rk_{AB}$  and Alice’s secret key  $sk_A$ , it should be impossible to derive any knowledge of Bob’s secret key  $sk_B$ .

**Unidirectionality in UE.** For UE schemes, there is an extra level of subtlety regarding the *directionality of keys* in addition to ciphertexts. A recent work of Jiang [Jia20] extensively studied the question: given an update token  $\Delta_e$  along with either  $k_{e-1}$  or  $k_e$ , is it possible to derive the other key? A scheme has bidirectional key updates if  $\Delta_e$  can be used to derive keys in both directions, and has unidirectional key updates if  $\Delta_e$  can be used in one direction, to derive  $k_e$  from  $k_{e-1}$ . Jiang [Jia20] showed that UE with bidirectional key and ciphertext updates implies UE with unidirectional key and ciphertext updates.

In the same work, Jiang postulated that to capture the same security level as the unidirectional PRE schemes, one requires even stronger UE schemes with *no-directional key updates*, where  $k_e$  cannot be derived from  $k_{e-1}$  and  $\Delta_e$ . In Jiang [Jia20], the definition of no-directional key updates intuitively requires that it is *also* impossible to derive  $k_{e-1}$  from  $\Delta_e$  and  $k_e$ . The recent work of Nishimaki [Nis21] proposed a seemingly weaker notion called *backward-leak unidirectional key updates* where  $\Delta_e$  can only be used in one direction to derive  $k_{e-1}$  from  $k_e$ . However, we observe that this new notion is essentially equivalent to no-directional key updates because derivation of  $k_{e-1}$  does not increase the adversary’s advantage in breaking the scheme. In particular, if the adversary obtains a ciphertext  $ct_{e-1}$  and corrupts  $\Delta_e$  and  $k_e$ , then it can first update the ciphertext to  $ct_e$  and decrypt it using  $k_e$ . Jiang emphasized that UE with no-directional key updates is the ideal security model, which by our argument above, extends to backwards-leak key updates. Henceforth, when we refer to unidirectional UE, we mean unidirectional UE with backwards-leak directional key updates unless otherwise specified.

**Gap between Unidirectionality and Bidirectionality.** In general, unidirectional UE and PRE schemes are more ideally suited to real-world applications as compared to their bidirectional counterparts due to their superior security guarantees. For example, unlike bidirectional UE schemes, unidirectional UE schemes guarantee security of data as if “freshly encrypted” in epoch  $e$  (i.e., not re-encrypted from epoch  $(e - 1)$ ) even if the adversary gains access to the secret key  $k_{e-1}$  and the update token  $\Delta_e$ . Unidirectional PRE schemes also offer similarly superior security guarantees over their bidirectional counterparts.

Another natural point of comparison between unidirectional and bidirectional UE and PRE schemes is the nature of cryptographic assumptions from which such schemes can be realized. Known constructions of UE and PRE seem-



ingly exemplify an apparent “gap” in terms of the assumptions required – unidirectional schemes have historically relied on stronger/more structured cryptographic assumptions as compared to their bidirectional counterparts.

Blaze et al. [BBS98] showed how to construct bidirectional PRE schemes from the Decisional Diffie-Hellman (DDH) assumption by suitably extending the well-known ElGamal encryption scheme [Gam85]. Similarly, a long line of works [BLMR13, LT18, KLR19, BDGJ20, Jia20] have shown how to realize bidirectional UE schemes as relatively simple extensions of public-key encryption from standard assumptions such as DDH and Learning With Errors (LWE).

On the other hand, unidirectional UE and PRE schemes typically rely on a stronger set of assumptions such as FHE [Gen09] and indistinguishability obfuscation (iO) [BGI+12], or highly structured cryptographic tools such as bilinear maps [BF03] and “hard” lattice trapdoors [GPV08]. Examples of constructions of unidirectional PRE from FHE and/or structured lattice trapdoors can be found in [NX15, CCL+14, Kir14, NAL15, PWA+16, FL17, PRSV17]. Constructions of unidirectional PRE schemes have also been shown to exist from bilinear maps [AFGH06, LV11]; however, these constructions are restricted to the *single-hop* setting in the sense that they only permit a single re-encryption of a ciphertext. Known constructions of unidirectional UE include the construction in [SS21] (which relies on bilinear maps), and two constructions in [Nis21] (one which achieves backward-leak key updates from lattice-specific techniques, and one which achieves no-directional key updates from iO). Sehrawat and Desmedt show a construction of UE from bi-homomorphic lattice-based pseudorandom functions [SD19]; however, their construction only achieves unidirectional ciphertext updates while still incurring bidirectional key updates (and is hence effectively bidirectional as per the recent findings in [Jia20]). To date, there exist no constructions of unidirectional PRE or UE from the plain DDH assumption (to our knowledge).

In this paper, we are motivated by the following longstanding open question in the study of UE and PRE:

*Do unidirectional UE and PRE schemes necessarily require stronger/more structured assumptions as compared to their bidirectional counterparts?*

More concretely, we ask the following question:

*Can we construct unidirectional UE/PRE schemes from DDH?*

## 1.1 Our Results

In this paper, we bridge this gap between the assumptions for unidirectional and bidirectional UE/PRE. We establish (somewhat surprisingly) that unidirectional UE and PRE schemes *do not*, in fact, require stronger/more structured cryptographic assumptions as compared to their bidirectional counterparts.

More concretely, we present generic constructions of unidirectional UE and PRE from a new primitive that we call Key and Plaintext Homomorphic Encryption (KPHE). We also show that such a KPHE scheme can be instantiated from

the BHHO encryption scheme [BHHO08] based on the DDH assumption. This yields the first constructions of unidirectional UE and PRE from the plain DDH assumption.

Our main result is summarized by the following (informal) theorem:

**Theorem 1 (Informal).** *Assuming the existence of a Key and Plaintext Homomorphic Encryption (KPHE) scheme that satisfies certain special properties, there exist post-compromise secure unidirectional UE and PRE schemes.*

**On KPHE.** The KPHE scheme with special properties required in our constructions can be viewed as a generalization of the BHHO public-key encryption scheme due to Boneh et al. [BHHO08]. It is a public key encryption scheme where the secret key is a bit-string  $\text{sk} \in \{0, 1\}^\ell$  and the plaintext is also a bit-string  $m \in \{0, 1\}^{\ell'}$  (in our constructions we use  $\ell = \ell' = 2n$ ). The specialized KPHE scheme satisfies the following three properties:

- **Distributional Semantic Security:** We require a KPHE scheme to achieve semantic security even when the secret keys are sampled from a specific distribution. In particular, we use KPHE schemes with  $2n$ -bit secret keys where the secret key is uniformly random subject to the constraint that it has equally many 0 and 1 bits (i.e.,  $n$  bits of 0 and  $n$  bits of 1).
- **Additive Key and Plaintext Homomorphisms:** We require a KPHE scheme to satisfy the following property: let  $T, T'$  be two arbitrary affine transformations that map 0–1 vectors to 0–1 vectors of the same length (in our constructions we use permutation maps over the bits of a  $2n$ -bit string). Then, given a public key  $\text{pk}$  corresponding to some secret key  $\text{sk}$  and a ciphertext  $\text{ct} \stackrel{\$}{\leftarrow} \text{Enc}(\text{pk}, m)$ , one can generate a public key  $\text{pk}'$  corresponding to the secret key  $T(\text{sk})$  and a ciphertext  $\text{ct}' \stackrel{\$}{\leftarrow} \text{Enc}(\text{pk}', T'(m))$ , without the knowledge of the original secret key  $\text{sk}$  or the original message  $m$ .
- **Blinding:** We also require the KPHE scheme to satisfy an associated security property called “blinding”, that (informally) argues that the public key and ciphertext generated via the aforementioned homomorphic transformations are indistinguishable from freshly generated public keys and ciphertexts (we make this more formal in Sect. 2).

For our PRE constructions, we also require that the KPHE scheme satisfies a notion of distributional circular security (i.e., circular security when the secret keys are sampled from a specific distribution). This is not required for our UE constructions.

**Instantiating KPHE.** We show how to concretely instantiate a KPHE scheme satisfying all of the aforementioned properties from DDH (based on the BHHO scheme [BHHO08]).

**Lemma 1 (Informal).** *Assuming DDH, there exists a secure construction of KPHE that satisfies the aforementioned properties.*

**Table 1.** Summary of bi/unidirectional UE and PRE schemes. We focus on ciphertext-independent UE and multi-hop PRE. In this table, “bi, uni, bwd-uni, no” stand for bidirectional, unidirectional, backward-leak unidirectional, no-directional, respectively. We note that the notion of *key-directionality* differs for UE and PRE; in the case of UE, unidirectionality of key updates implies that, given the source (secret) key and the update token, the destination (secret) key can be computed. This is not the case for PRE, where unidirectional key update simply denotes that the re-key generation algorithm takes as input the source secret key and the destination public key (as opposed to bidirectional key update, where the re-key generation algorithm takes as input both secret keys).

	scheme	dir. (ctx)	dir. (key)	security	assumption
UE	[BLMR13]	bi	bi	IND-ENC	DDH/LWE
UE	[LT18, KLR19, BDGJ20]	bi	bi	IND-UE	DDH
UE	[Jia20]	bi	bi	IND-UE	DLWE
UE	[SD19]	uni	bi	IND-UE	LWE
UE	[Nis21]	uni	bwd-uni	IND-UE	LWE
UE	[Nis21]	uni	no	IND-UE	iO
UE	[SS21]	uni	no	IND-UE	SXDH
UE	Ours	uni	bwd-uni	IND-UE	DDH
PRE	[BBS98]	bi	bi	IND-CPA	DDH
PRE	[CCL+14]	uni	uni	IND-CPA	DLWE
PRE	[PWA+16]	uni	uni	IND-CCA	LWE
PRE	[PRSV17]	uni	uni	IND-CPA	RLWE
PRE	Ours	uni	uni	IND-HRA	DDH

**Corollary 1 (Informal).** *Assuming DDH, there exist post-compromise secure unidirectional UE and PRE schemes.*

**Security of Our Constructions.** Our constructions of unidirectional UE and PRE achieve the strongest notions of *post-compromise security* in the standard model. Our construction of unidirectional UE achieves the state-of-the-art post-compromise security definition due to Boyd et al. [BDGJ20], while also ensuring backward-leak unidirectional key updates [Nis21]. Our unidirectional PRE construction achieves the post-compromise security definition recently proposed by Davidson et al. [DDL19], which is, to our knowledge, the only notion of post-compromise PRE security to be proposed to date. We present a more detailed discussion on post-compromise security (and other related security notions) of UE and PRE in the next subsection. Table 1 presents a comparison of our results with those in the existing literature.

## 1.2 Background and Related Work

There has been extensive research on both UE and PRE, including various settings, definitions, and constructions. Below we only mention works that are the

most directly relevant. For both UE and PRE, we focus on the CPA-type definitions, which are by far the most well-studied notions.

**Security Notions for UE.** Since the introduction of UE in [BLMR13], several works have explored its security notions [EPRS17,LT18,AMP19,KLR19,BDGJ20,Jia20]. Most notable is the work of Lehmann and Tackmann [LT18], which improved the model and studied the notion of post-compromise security for UE. Their Indistinguishability of Update notion (IND-UPD) returns a challenge ciphertext  $ct^*$  which is either the re-encryption of a ciphertext  $ct_0$  or  $ct_1$ . A scheme is IND-UPD secure if an adversary is unable to determine which of the ciphertexts was re-encrypted.

In subsequent works a stronger combined notion of IND-UE security has been used, first defined by Boyd et al. [BDGJ20]. The IND-UE notion requires an adversary be unable to distinguish between a fresh encryption of a plaintext  $m$  and the re-encryption of a ciphertext  $ct$ . As a result this notion captures both CPA (specifically IND-ENC) and IND-UPD security.

**Security Notions for PRE.** In the context of PRE, the traditional notion of IND-CPA security [ID03,AFGH06] have been shown to be insufficient in practice. To address this, Cohen [Coh19] introduced the notion of Honest Re-Encryption Attack (HRA) security where an adversary is additionally permitted to re-encrypt (from honest to corrupt users) ciphertexts previously output by the encryption oracle. While only recently considered in the analysis of PRE, the essence of this notion is also fundamental in formalizing security for UE.

More recently, Davidson et al. [DDL19] have investigated achieving post-compromise secure PRE schemes. They introduced a notion of IND-PCS security for PRE, which can be viewed as the analogue of IND-UPD security of UE in the context of PRE, albeit for more complex re-encryption graphs. To date this is the only paper that studies the PCS security of PRE schemes. Their work again demonstrates the challenges in constructing such schemes in the unidirectional setting. They discuss two PCS-secure constructions which are based on a prior unidirectional PRE scheme, Construction 7b of Fuchsbauer et al. [FKKP19] and an extension of BV-PRE [PRSV17].

**Updatable Public Key Encryption.** In order to achieve forward security in public key encryption (PKE), a notion called *updatable PKE (UPKE)* has recently been proposed and studied [JMM19,ACDT20,DKW21], where any sender (encryptor) can initiate a key update by sending a special update ciphertext to the receiver (decryptor). This ciphertext updates the public key and also, once processed by the receiver, will update its secret key. These are PKE schemes that encrypt messages under different public keys and aim to achieve forward security. In contrast, UE and PRE schemes studied in this paper aim to update ciphertexts encrypted under an old key to a new key without leaking the message content. The notions of UE/PRE as well as our techniques are very different from UPKE despite the partial naming collision.

**Comparison with Umbral.** There exists a practically deployed construction of unidirectional PRE, namely Umbral [Nun18], from the DDH Assumption, albeit

in the random oracle model. It turns out that the Umbral construction is only single-hop, and focuses on achieving threshold PRE rather than multi-hop PRE. In particular, the Umbral construction crucially relies on the Diffie-Hellman key change, and it is unclear how to extend the construction to multiple hops. On the other hand, our primary aim is to achieve multi-hop unidirectional PRE in the traditional non-threshold setting. We note additionally that Umbral would not achieve post-compromise security, which is an important property provided by our constructions. Fundamentally, this is due to the fact that Umbral adopts a KEM-DEM style approach where only the KEM is re-encrypted.

**Concurrent Work.** A concurrent work by Galteland and Pan [GP23] constructs unidirectional UE with backward-leak unidirectional key update from public key encryption (PKE) schemes with certain properties, which can be realized from the DDH or LWE assumption. Their techniques are significantly different from ours and do not trivially extend to the PRE setting. The authors of [GP23] also demonstrate a formal proof that the security definition for unidirectional UE with backward-leak unidirectional key updates is equivalent to the one with no-directional key updates, which confirms our observation discussed earlier.

### 1.3 Technical Overview

In this section, we provide a high-level overview of our techniques for constructing unidirectional UE and PRE from any generic KPHE scheme satisfying the special properties described earlier.

**IND-ENC Secure UE.** Our first attempt is to build a unidirectional IND-ENC secure UE scheme, and we start with a naïve idea. Take an arbitrary symmetric-key encryption scheme and each epoch key is a freshly generated key of this encryption scheme. The update token  $\Delta_e$  from  $\mathbf{k}_{e-1}$  to  $\mathbf{k}_e$  is an encryption of  $\mathbf{k}_{e-1}$  under  $\mathbf{k}_e$ , namely  $\Delta_e = \text{Enc}_{\mathbf{k}_e}(\mathbf{k}_{e-1})$ . When we update a ciphertext from epoch  $(e-1)$  to epoch  $e$ , we just attach the update token  $\Delta_e$  to the end of the ciphertext. For a message  $m$  first encrypted in epoch  $e$  and then updated through epoch  $e'$ , the resulting ciphertext is of the form:

$$\text{ct}_{e'} = (\text{Enc}_{\mathbf{k}_e}(m), \text{Enc}_{\mathbf{k}_{e+1}}(\mathbf{k}_e), \dots, \text{Enc}_{\mathbf{k}_{e'}}(\mathbf{k}_{e'-1})).$$

Given  $\mathbf{k}_{e'}$ , one can easily decrypt  $\text{ct}_{e'}$  layer by layer to recover  $m$ .

This naïve approach does not achieve IND-ENC security. We show a concrete attack in the following. Let  $e^*$  be the challenge epoch and  $m^*$  be the challenge message queried by the adversary. Let  $\text{ct}_{e^*} = \text{Enc}_{\mathbf{k}_{e^*}}(m^*)$  be the challenge ciphertext. To extract the secret key  $\mathbf{k}_{e^*}$ , the adversary proceeds as follows. It first queries for an encryption of an arbitrary message  $m$  in epoch 0 and then updates it to epoch  $e$  (for some  $e > e^*$ ) via a sequence of update queries. This way the adversary obtains a ciphertext of  $m$  of the form:

$$\text{ct}_e = (\text{Enc}_{\mathbf{k}_0}(m), \text{Enc}_{\mathbf{k}_1}(\mathbf{k}_0), \dots, \text{Enc}_{\mathbf{k}_e}(\mathbf{k}_{e-1})).$$

Now the adversary corrupts the secret key  $k_e$ . Then it can recover all the previous keys from  $k_0$  to  $k_{e-1}$  (including  $k_{e^*}$ ) during decryption of the ciphertext  $ct_e$ .

Nonetheless, this simple approach demonstrates some nice properties of unidirectionality. For key updates, it is impossible to derive  $k_e$  from  $k_{e-1}$  and  $\Delta_e$ . For ciphertext updates, given a *fresh* ciphertext  $ct_e$  in epoch  $e$  and the previous update token  $\Delta_e$  (from epoch  $(e - 1)$  to  $e$ ), it is impossible to transition the ciphertext  $ct_e$  to the previous epoch  $ct_{e-1}$  (i.e. the epoch prior to its existence). In fact, Cohen [Coh19] applied this idea to PRE and showed a CPA-secure but not HRA-secure PRE scheme (HRA security is inherently required in IND-ENC UE schemes).

**Re-randomizing the Secret Keys.** The problem with these chained ciphertexts is that during decryption of a single ciphertext, all the previous secret keys are also leaked. To resolve this problem, our hope is to somehow re-randomize all the previous secret keys in the chain, in a consistent and homomorphic manner. In particular, we want the ciphertext to be of the form

$$ct_e = \left( \text{Enc}_{\bar{k}_0}(m), \text{Enc}_{\bar{k}_1}(\bar{k}_0), \dots, \text{Enc}_{\bar{k}_{e-1}}(\bar{k}_{e-2}), \text{Enc}_{k_e}(\bar{k}_{e-1}) \right),$$

where  $\bar{k}_0, \bar{k}_1, \dots, \bar{k}_{e-1}$  are all re-randomized secret keys that are different for each ciphertext. During the decryption of  $ct_e$ , only these re-randomized secret keys are leaked, which does not affect the security of other ciphertexts.

To enable such re-randomization, our idea is inspired by the re-randomizable Yao’s garbled circuits [GHV10]. We propose a new primitive called Key and Plaintext Homomorphic Encryption (KPHE), which can be seen as a generalization of the circular secure encryption scheme of Boneh et al. [BHHO08]. Instead of using an arbitrary symmetric-key encryption scheme, we use the KPHE scheme for encryption, where the UE secret key  $k_e$  is a key pair  $(pk_e, sk_e)$  of the KPHE scheme. The update token is a KPHE encryption of the previous epoch’s secret key under the current epoch’s public key, namely  $\Delta_e = \text{KPHE.Enc}_{pk_e}(sk_{e-1})$ .

To update a ciphertext we exploit the two homomorphism properties of the KPHE scheme, in both the message space and the key space. Given an update token  $\Delta_e$  and a ciphertext of the form

$$ct_{e-1} = \left( \text{KPHE.Enc}_{\bar{pk}_0}(m), \text{KPHE.Enc}_{\bar{pk}_1}(\bar{sk}_0), \dots, \text{KPHE.Enc}_{\bar{pk}_{e-1}}(\bar{sk}_{e-2}) \right),$$

we focus on the last component  $ctx = \text{KPHE.Enc}_{\bar{pk}_{e-1}}(\bar{sk}_{e-2})$  and the update token  $\Delta_e = \text{KPHE.Enc}_{pk_e}(sk_{e-1})$ . In our update operation we first generate a random permutation  $\pi$  and then perform two important steps:

- Use the KPHE key-space homomorphism to transform  $ctx$  from an encryption under  $sk_{e-1}$  to an encryption under  $\pi(sk_{e-1})$ .
- Use the KPHE message-space homomorphism to transform  $\Delta_e$  from an encryption of  $sk_{e-1}$  to an encryption of  $\pi(sk_{e-1})$ .

The updated ciphertext becomes

$$ct_e = \left( \text{KPHE.Enc}_{\overline{pk}_0}(m), \text{KPHE.Enc}_{\overline{pk}_1}(\overline{sk}_0), \dots, \text{KPHE.Enc}_{\overline{pk}_{e-1}}(\overline{sk}_{e-2}), \text{KPHE.Enc}_{pk_e}(\overline{sk}_{e-1}) \right),$$

where  $\overline{sk}_{e-1} = \pi(sk_{e-1})$  (with corresponding public key  $\overline{pk}_{e-1}$ ). In our construction, the KPHE secret key is a  $2n$ -bit string, which is randomly sampled with exactly  $n$  bits of 0 and  $n$  bits of 1. The affine transformation  $\pi$  is a random permutation on the  $2n$  bits of the string. By transforming from  $sk_{e-1}$  to  $\overline{sk}_{e-1}$  we ensure that a fresh secret key is used for each update operation and hence there is appropriate isolation between all ciphertexts updated in a given epoch. The blinding property of KPHE ensures that re-randomization can be done without knowledge of the underlying secret keys, and that the re-randomized ciphertexts are computationally indistinguishable from freshly generated ciphertexts.

**Use of Balanced KPHE Keys.** The astute reader might have noticed that we use “balanced” secret keys for our KPHE scheme, wherein each secret key is a randomly sampled  $2n$ -bit string with exactly  $n$  bits of 0 and  $n$  bits of 1. The restriction is required to offset some leakage that our scheme incurs during the honest re-encryption query phase in the security proofs. Informally, the adversary can use a sequence of honest re-encryption queries to learn some information about the intermediate (re-randomized) secret keys; in particular, it learns the number of 0 and 1 bits in each secret key. Intuitively, we offset this leakage by specifying at setup that all secret keys have an equal number of 0 and 1 entries. As a result, the adversary learns no additional information about these intermediate keys, irrespective of the number of honest re-encryption queries that it issues. We defer a formal treatment to the detailed proofs of security for our constructions.

**Achieving Post-Compromise Secure UE.** We can extend the IND-ENC secure UE construction to achieve post-compromise security. To achieve IND-UPD security, we can modify the update operation to ensure that all the chained ciphertexts are updated (rather than just the last one). In effect what our enhanced construction does is again exploit properties of the KPHE scheme to re-randomize each of the ciphertext components. This ensures that two updated ciphertexts of the same length are computationally indistinguishable. To further achieve the combined IND-UE security, we need to additionally guarantee that a freshly generated ciphertext has the same length as an updated ciphertext in a certain epoch. More details on our UE constructions are given in Sect. 3.

**Achieving Unidirectional PRE.** We can use the same high-level approach to construct a unidirectional PRE scheme, where a ciphertext consists of a chain of KPHE ciphertexts, and re-encryption exploits the two KPHE homomorphisms to transform each new KPHE ciphertext to a fresh secret key. The crucial subtlety in the PRE case, which makes proving security slightly more involved, is that we no longer consider sequential ciphertext updates but must consider re-encryption between all possible key pairs. As a result we need to further exploit the circular security properties of the KPHE scheme to prove security. This is further detailed in Sect. 4.

**Connections between UE and PRE.** Generally speaking, unidirectional PRE can be viewed as a stronger primitive than unidirectional UE because UE only allows for sequential updates while PRE allows for re-encryption between every pair of keys. In fact, we observe that if we treat the public-secret key pair of PRE as a secret key for UE, and the PRE re-encryption key as an update token for UE, then IND-HRA secure PRE implies IND-ENC secure UE, and IND-PCS secure PRE implies IND-UPD secure UE. This is also why our constructions for unidirectional UE and PRE follow a very similar framework. On the other hand, since PRE supports re-encryption between (potentially) every pair of keys, our constructions of PRE require stronger security guarantees (in particular, circular security) from the underlying KPHE scheme.

**Efficiency and Feasibility.** We acknowledge that the ciphertext length in our UE/PRE constructions grows linearly with the number of epochs/re-encryption hops, unlike certain existing constructions (e.g. in [Nis21, PWA+16, PRSV17]) where the ciphertext size remains the same. In this context, we emphasize that our paper is the first to achieve backward-leak unidirectional UE and unidirectional PRE from standard assumptions, specifically DDH. It has been a long-standing open problem for over a decade whether obfuscation/FHE is necessary for unidirectional UE/(multi-hop) PRE, and our work closes this assumption gap. As a result, we believe that our results should be viewed with emphasis on the new theoretical insights/understanding into unidirectional-UE/PRE that they enable as opposed to concrete efficiency. Our work opens up the discussion of whether obfuscation/FHE is necessary for achieving unidirectional UE/PRE with “succinctness” in the ciphertext length.

We note that in the UE setting, key rotation may only happen a small number of times in practice. For example, once a year for the lifetime of the ciphertext (say 10years). Thus, taking a similar approach to [BEKS20] (from the ciphertext-dependent setting) we could bound the number of updates and have fixed-length ciphertexts through some form of padding. We also point out that while the size of ciphertexts in our general constructions grow linearly, the secret keys and update tokens/re-encryption keys remain constant-sized. We also note that for the basic versions of our UE/PRE construction (IND-CPA unidirectional UE and IND-HRA secure unidirectional PRE), the work done per update/re-encryption operation is also constant (independent of the number of epochs/update hops).

We note here that a naïve approach to achieving unidirectional UE is the so called “download–decrypt–re-encrypt–upload” approach, where the client downloads the encrypted data (e.g. from the server storing the encrypted data), locally decrypts it, re-encrypts it using the new key, and re-uploads the newly encrypted data to the server. Our UE constructions are non-trivial in the sense that we achieve significantly better properties as compared to this naïve approach. In particular, for applications of UE (e.g. key rotation) where the client outsources encrypted data to the server, this entails constant computational/ communication/storage overheads at the client during key rotation (the client simply



generates and sends the update token to the server); the corresponding client-overheads are linear (in database size) in the naïve solution.

**Using Random Oracles.** A possible approach towards achieving practical efficiency is to use random oracles (such as in the single-hop unidirectional threshold PRE scheme Umbral [Nun18]). Our focus is primarily on feasibility results for unidirectional UE/PRE in the standard model, and we consciously avoid the use of random oracles. We also point out that a previous result [AMP19] showed that, even in the symmetric-key setting, unidirectional UE/PRE implies public-key encryption, and so a construction from just a random oracle is unlikely. However, it might be possible to achieve efficiency gains using a random oracle. We leave investigating such a random oracle-based construction of unidirectional UE/PRE as an interesting direction of future research.

## 1.4 Paper Outline

The rest of the paper is organized as follows. Section 2 formally defines a KPHE scheme and its associated security properties. Section 3 presents our constructions of IND-CPA and IND-UPD secure UE from any KPHE scheme. Section 4 presents our construction of IND-HRA secure PRE from any KPHE scheme. We defer detailed proofs of security for these schemes, our constructions of IND-UE secure UE and IND-PCS secure PRE from any KPHE scheme, and the instantiation of KPHE from DDH to the full version of our paper [MPW22].

For readers not familiar with the formal definitions of UE and PRE, we present relatively self-contained background material on UE and PRE in Sects. 3.1 and 4.1, respectively. Due to lack of space, the formal security notions of PRE are deferred to the full version of our paper [MPW22].

## 1.5 Notations

We summarize here the notations used in the rest of the paper. We write  $x \stackrel{\$}{\leftarrow} \mathcal{X}$  to represent that an element  $x$  is sampled randomly from a set/distribution  $\mathcal{X}$ . The output  $x$  of a deterministic (resp. randomized) algorithm  $\mathcal{A}$  is denoted by  $x = \mathcal{A}$  (resp.  $x \stackrel{\$}{\leftarrow} \mathcal{A}$ ). For  $a \in \mathbb{N}$  such that  $a \geq 1$ , we denote by  $[a]$  the set of integers lying between 1 and  $a$  (both inclusive). We refer to  $\lambda \in \mathbb{N}$  as the security parameter, and denote by  $\text{poly}(\lambda)$  and  $\text{negl}(\lambda)$  any generic (unspecified) polynomial function and negligible function in  $\lambda$ , respectively.

## 2 Key and Plaintext Homomorphic Encryption

In this section, we present the definitions for the core building block for our constructions, namely key and plaintext homomorphic encryption (KPHE). Informally, a KPHE scheme has the following features:

- **Keys and Plaintexts:** Each secret key  $\text{sk}$  is an  $\ell$ -bit string for some  $\ell = \text{poly}(\lambda)$  ( $\lambda$  being the security parameter). Additionally, each plaintext message  $\text{m}$  is an  $\ell'$ -bit string for some  $\ell' = \text{poly}(\lambda)$ .
- **Key Distribution:** Each secret key is sampled according to some distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$ . In particular, for our applications, we assume KPHE schemes where each secret key  $\text{sk}$  is a  $2n$ -bit string with equally many 0 and 1 entries.
- **Key Homomorphism:** Let  $T$  be any linear transformation that maps  $\ell$ -bit strings to  $\ell$ -bit strings. Then, it is possible to efficiently evaluate the following:
  - Given a public key  $\text{pk}$  corresponding to some secret key  $\text{sk} \in \{0, 1\}^\ell$ , it is possible to efficiently compute a valid public key  $\text{pk}'$  corresponding to the transformed secret key  $\text{sk}' = T(\text{sk})$ , without the knowledge of  $\text{sk}$ .
  - Given a ciphertext  $\text{ct}$  encrypting a message  $\text{m}$  under some secret key  $\text{sk} \in \{0, 1\}^\ell$ , it is possible to efficiently compute a ciphertext  $\text{ct}'$  encrypting the same message  $\text{m}$  under the transformed secret key  $\text{sk}' = T(\text{sk})$ , without the knowledge of  $\text{sk}$ .
- **Plaintext Homomorphism:** Let  $T'$  be any linear transformation that maps  $\ell'$ -bit strings to  $\ell'$ -bit strings. Then, given a ciphertext  $\text{ct}$  encrypting a message  $\text{m} \in \{0, 1\}^{\ell'}$  under some secret key  $\text{sk}$ , it is possible to efficiently compute a ciphertext  $\text{ct}''$  encrypting the transformed message  $\text{m}' = T'(\text{m})$  under the same secret key  $\text{sk}$ .

We now summarize these features of KPHE formally below.

**Definition 1 (KPHE).** *A KPHE scheme is a tuple of PPT algorithms of the form  $\text{KPHE} = (\text{Setup}, \text{SKGen}, \text{PKGen}, \text{Enc}, \text{Dec}, \text{Eval})$  that are defined as follows:*

- $\text{pp} \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda)$ : *On input the security parameter  $\lambda$ , the setup algorithm outputs a public parameter  $\text{pp}$ .*
- $\text{sk} \stackrel{\$}{\leftarrow} \text{SKGen}(\text{pp}, \mathcal{D})$ : *On input the public parameter  $\text{pp}$  and a distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  (for  $\ell = \text{poly}(\lambda)$ ), the secret key generation algorithm outputs a secret key  $\text{sk} \stackrel{\$}{\leftarrow} \mathcal{D}$ .*
- $\text{pk} \stackrel{\$}{\leftarrow} \text{PKGen}(\text{pp}, \text{sk})$ : *On input the public parameter  $\text{pp}$  and a secret key  $\text{sk} \in \{0, 1\}^\ell$ , the public key generation algorithm outputs a public key  $\text{pk}$ .*
- $\text{ct} \stackrel{\$}{\leftarrow} \text{Enc}(\text{pk}, \text{m})$ : *On input a public key  $\text{pk}$  and a message  $\text{m} \in \{0, 1\}^{\ell'}$  (for  $\ell' = \text{poly}(\lambda)$ ), the encryption algorithm outputs a ciphertext  $\text{ct}$ .*
- $\text{m}/\perp \stackrel{\$}{\leftarrow} \text{Dec}(\text{sk}, \text{ct})$ : *On input a secret key  $\text{sk} \in \{0, 1\}^\ell$  and a ciphertext  $\text{ct}$ , the decryption algorithm outputs a plaintext message string  $\text{m}$  or an error symbol  $\perp$ .*
- $(\text{pk}', \text{ct}') \stackrel{\$}{\leftarrow} \text{Eval}(\text{pk}, \text{ct}, T, T')$ : *On input a public key  $\text{pk}$ , a ciphertext  $\text{ct}$ , and a pair of (linear) transformations  $T : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$  and  $T' : \{0, 1\}^{\ell'} \rightarrow \{0, 1\}^{\ell'}$ , the homomorphic evaluation algorithm outputs a tuple consisting of a transformed public key and a transformed ciphertext  $(\text{pk}', \text{ct}')$ .*

**Correctness.** A KPHE scheme ( $\text{Setup}, \text{SKGen}, \text{PKGen}, \text{Enc}, \text{Dec}, \text{Eval}$ ) is said to be correct with respect to a distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  if for any  $\text{pp} \xleftarrow{\$} \text{Setup}(1^\lambda)$ , any  $\text{sk} \xleftarrow{\$} \text{SKGen}(\text{pp}, \mathcal{D})$ , any  $\text{pk} \xleftarrow{\$} \text{PKGen}(\text{pp}, \text{sk})$ , any  $\text{m} \in \{0, 1\}^{\ell'}$ , and any pair of (linear) transformations  $T : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$  and  $T' : \{0, 1\}^{\ell'} \rightarrow \{0, 1\}^{\ell'}$ , letting  $\text{sk}' = T(\text{sk})$ ,  $\text{m}' = T'(\text{m})$  and

$$\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, \text{m}), \quad (\text{pk}', \text{ct}') \xleftarrow{\$} \text{Eval}(\text{pk}, \text{ct}, T, T'),$$

both of the following hold with overwhelmingly large probability:

- $\text{pk}'$  is a valid public key with respect to  $\text{sk}' = T(\text{sk})$ , i.e., for any  $\bar{\text{m}} \in \{0, 1\}^{\ell'}$ , it holds that  $\text{Dec}(\text{sk}', \text{Enc}(\text{pk}', \bar{\text{m}})) = \bar{\text{m}}$ .
- $\text{ct}'$  is a valid encryption of  $\text{m}'$  under  $(\text{pk}', \text{sk}')$ , i.e.,  $\text{Dec}(\text{sk}', \text{ct}') = \text{m}'$ .

**Distributional Semantic Security.** We (informally) say that a KPHE satisfies distributional semantic security with respect to some distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  if it remains semantically secure even when the secret key  $\text{sk}$  is sampled according to the distribution  $\mathcal{D}$ . Formally, this is modeled using a semantic security game where the secret key is sampled by the challenger as per the distribution  $\mathcal{D}$ .

**Definition 2 ( $\mathcal{D}$ -Semantic Security).** A KPHE scheme with  $\ell$ -bit secret keys is said to be  $\mathcal{D}$ -semantically secure with respect to a distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  if for any security parameter  $\lambda \in \mathbb{N}$  and any PPT adversary  $\mathcal{A}$ , the following holds with overwhelmingly large probability:

$$| \Pr[\text{Expt}_{\mathcal{D}}^{\text{DSS-KPHE}}(\lambda, \mathcal{A}) = 1] - 1/2 | < \text{negl}(\lambda),$$

where the experiment  $\text{Expt}_{\mathcal{D}}^{\text{DSS-KPHE}}(\lambda, \mathcal{A})$  is as defined in Fig. 1.

**Distributional Circular Security.** We (informally) say that a KPHE satisfies distributional circular security with respect to some distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  if it satisfies the standard notion of circular security [CL01, BRS02, BH08, ACPS09] even when each secret key is sampled from the distribution  $\mathcal{D}$ . Formally, this is modeled using a circular security game where the secret keys are sampled by the challenger as per the distribution  $\mathcal{D}$ .

**Definition 3 ( $\mathcal{D}$ -Circular Security).** A KPHE scheme with  $\ell$ -bit secret keys and  $\ell$ -bit messages is said to be  $\mathcal{D}$ -circular secure with respect to a distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  if for any security parameter  $\lambda \in \mathbb{N}$  and any PPT adversary  $\mathcal{A}$ , the following holds with overwhelmingly large probability:

$$| \Pr[\text{Expt}_{\mathcal{D}}^{\text{DCC-KPHE}}(\lambda, \mathcal{A}) = 1] - 1/2 | < \text{negl}(\lambda),$$

where the experiment  $\text{Expt}_{\mathcal{D}}^{\text{DCC-KPHE}}(\lambda, \mathcal{A})$  is as defined in Fig. 2.

**Experiment**  $\text{Expt}_{\mathcal{D}}^{\text{DSS-KPHE}}(\lambda, \mathcal{A})$ :

1. The challenger generates  $\text{pp} \xleftarrow{\$} \text{Setup}(1^\lambda)$ ,  $\text{sk} \xleftarrow{\$} \text{SKGen}(\text{pp}, \mathcal{D})$ , and  $\text{pk} \xleftarrow{\$} \text{PKGen}(\text{pp}, \text{sk})$ , and provides the adversary  $\mathcal{A}$  with  $(\text{pp}, \text{pk})$ .
2. The adversary  $\mathcal{A}$  issues a challenge encryption query for a pair of messages  $(m_0, m_1)$ .
3. The challenger samples  $b \xleftarrow{\$} \{0, 1\}$ , creates the challenge ciphertext

$$\text{ct}^* \xleftarrow{\$} \text{Enc}(\text{pk}, m_b),$$

and sends  $\text{ct}^*$  to the adversary  $\mathcal{A}$ .

4. The adversary  $\mathcal{A}$  outputs a bit  $b' \in \{0, 1\}$ .
5. Output 1 if  $b = b'$  and 0 otherwise.

**Fig. 1.** The  $\mathcal{D}$ -Semantic Security Experiment for KPHE

**Experiment**  $\text{Expt}_{\mathcal{D}}^{\text{DCC-KPHE}}(\lambda, \mathcal{A})$ :

1. The challenger generates  $\text{pp} \xleftarrow{\$} \text{Setup}(1^\lambda)$  and provides it to the adversary.
2. The adversary  $\mathcal{A}$  outputs  $n = \text{poly}(\lambda)$ .
3. The challenger samples  $\text{sk}_1, \dots, \text{sk}_n \xleftarrow{\$} \text{SKGen}(\text{pp}, \mathcal{D})$ , sets

$$\text{pk}_1 \xleftarrow{\$} \text{PKGen}(\text{pp}, \text{sk}_1), \dots, \text{pk}_n \xleftarrow{\$} \text{PKGen}(\text{pp}, \text{sk}_n),$$

and provides  $(\text{pk}_1, \dots, \text{pk}_n)$  to the adversary  $\mathcal{A}$ .

4. The challenger also sets the following for each  $i, j \in [n]$ :

$$\text{ct}_{i,j,0} \xleftarrow{\$} \text{Enc}(\text{pk}_i, \text{sk}_j), \quad \text{ct}_{i,j,1} \xleftarrow{\$} \text{Enc}(\text{pk}_i, 0^{|\text{sk}_j|}).$$

5. The challenger finally samples a bit  $b \xleftarrow{\$} \{0, 1\}$  and provides the adversary  $\mathcal{A}$  with the ensemble  $\{\text{ct}_{i,j,b}\}_{i,j \in [n]}$ .
6. The adversary  $\mathcal{A}$  outputs a bit  $b' \in \{0, 1\}$ .
7. Output 1 if  $b = b'$  and 0 otherwise.

**Fig. 2.** The  $\mathcal{D}$ -Circular Security Experiment for KPHE

**Blinding.** We (informally) say that a KPHE scheme satisfies public key and ciphertext blinding if the homomorphic evaluation algorithm outputs a public key-ciphertext pair  $(\text{pk}', \text{ct}')$  corresponding to the transformed secret key  $\text{sk}'$  and the transformed message  $m'$  such that:

- The transformed public key  $\text{pk}'$  is computationally indistinguishable from a public key sampled uniformly at random from the set of all valid public keys corresponding to the secret key  $\text{sk}'$ .
- The transformed ciphertext  $\text{ct}'$  is computationally indistinguishable from a ciphertext sampled uniformly at random from the set of all valid ciphertexts corresponding to the transformed message  $m'$  under  $\text{pk}'$ .

**Experiment**  $\text{Expt}_{\mathcal{D}}^{\text{Blind-KPHE}}(\lambda, \mathcal{A})$ :

1. The challenger generates  $\text{pp} \xleftarrow{\$} \text{Setup}(1^\lambda)$ ,  $\text{sk} \xleftarrow{\$} \mathcal{D}$ , and  $\text{pk} \xleftarrow{\$} \text{PKGen}(\text{pp}, \text{sk})$ , and provides the adversary  $\mathcal{A}$  with  $(\text{pp}, \text{sk}, \text{pk})$ .
2. The adversary  $\mathcal{A}$  sends a message  $m \in \{0, 1\}^{\ell'}$  to the challenger.
3. The challenger responds to  $\mathcal{A}$  with a ciphertext  $\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, m)$ .
4. The adversary  $\mathcal{A}$  then sends a pair of (linear) transformations
 
$$T : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell, \quad T' : \{0, 1\}^{\ell'} \rightarrow \{0, 1\}^{\ell'}.$$
5. The challenger sets
 
$$\text{sk}' = T(\text{sk}), \quad m' = T'(m),$$
 and computes the following:
 
$$(\text{pk}_0, \text{ct}_0) \xleftarrow{\$} \text{Eval}(\text{pk}, \text{ct}, T, T'), \quad \text{pk}_1 \xleftarrow{\$} \text{PKGen}(\text{pp}, \text{sk}'), \quad \text{ct}_1 \xleftarrow{\$} \text{Enc}(\text{pk}_1, m'),$$
6. The challenger finally samples a bit  $b \xleftarrow{\$} \{0, 1\}$  and provides the adversary  $\mathcal{A}$  with  $(\text{pk}_b, \text{ct}_b)$ .
7. The adversary  $\mathcal{A}$  outputs a bit  $b' \in \{0, 1\}$ .
8. Output 1 if  $b = b'$  and 0 otherwise.

**Fig. 3.** The Blinding Experiment for KPHE

More formally, we define this blinding property as follows.

**Definition 4 (Blinding).** *A KPHE scheme with  $\ell$ -bit secret keys and  $\ell'$ -bit messages is said to satisfy blinding security with respect to a distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$  if for any security parameter  $\lambda \in \mathbb{N}$  and any PPT adversary  $\mathcal{A}$ , the following holds with overwhelmingly large probability:*

$$| \Pr[\text{Expt}_{\mathcal{D}}^{\text{Blind-KPHE}}(\lambda, \mathcal{A}) = 1] - 1/2 | < \text{negl}(\lambda),$$

where the experiment  $\text{Expt}_{\mathcal{D}}^{\text{Blind-KPHE}}(\lambda, \mathcal{A})$  is as defined in Fig. 3.

**KPHE from DDH.** In full version of our paper [MPW22], we prove the following (informal) theorem:

**Theorem 2 (Informal).** *Assuming DDH, there exists a KPHE scheme with  $2n$ -bit secret keys that satisfies distributional semantic security with respect to the distribution  $\mathcal{U}_n$ , distributional circular security with respect to the distribution  $\mathcal{U}_n$ , and blinding, as defined above.*

In particular, we rely on known results from [BHHO08, NS12, GHV10] for the DDH-based instantiation of KPHE. See [MPW22] for details.

**KPHE from LWE.** In this paper, we do not explicitly describe a construction of KPHE from LWE since there already exist constructions of unidirectional

UE/PRE from LWE [CCL+14, PWA+16, PRSV17, Nis21]. Our aim in this work is to close the gap between bidirectional and unidirectional constructions of UE/PRE in terms of assumptions, and so we choose to focus on the feasibility results from the DDH assumption.

We note, however, that constructing KPHE from LWE is a very interesting direction of future work. In particular, one needs to be careful during re-encryption, which potentially increases the level of noise in the ciphertext of the LWE-based encryption scheme and could leak extra information. For example, due to increase in the noise level during re-encryption, it is not straightforward to prove the ciphertext blinding property, which requires that a freshly created ciphertext and a re-encrypted ciphertext are distributed in an indistinguishable manner. This issue can be handled using noise flooding techniques, albeit at the cost of a larger ciphertext size.

**KPHE from Other Assumptions.** We also leave it as an interesting open question to construct KPHE from concrete hardness assumptions other than DDH or LWE (e.g., factorization-based assumptions or LPN). Given our results on achieving unidirectional UE/PRE from KPHE, such realizations of KPHE would immediately yield new constructions of unidirectional UE/PRE from these assumptions.

### 3 Unidirectional UE from KPHE

In this section, we show how to construct unidirectional UE satisfying various security notions (IND-ENC, IND-UPD and IND-UE) from any KPHE scheme.

#### 3.1 Definition

**Definition 5.** *An updatable encryption (UE) scheme for message space  $\mathcal{M}$  is a tuple of PPT algorithms  $\text{UE} = (\text{UE.setup}, \text{UE.next}, \text{UE.enc}, \text{UE.upd}, \text{UE.dec})$  with the following syntax:*

- $\mathbf{k}_0 \stackrel{\$}{\leftarrow} \text{UE.setup}(1^\lambda)$ : On input a security parameter  $1^\lambda$ , it returns a secret key  $\mathbf{k}_e$  for epoch  $e = 0$ .
- $(\mathbf{k}_{e+1}, \Delta_{e+1}) \stackrel{\$}{\leftarrow} \text{UE.next}(\mathbf{k}_e)$ : On input a secret key  $\mathbf{k}_e$  for epoch  $e$ , it outputs a new secret key  $\mathbf{k}_{e+1}$  and an update token  $\Delta_{e+1}$  for epoch  $e + 1$ .
- $\text{ct}_e \stackrel{\$}{\leftarrow} \text{UE.enc}(\mathbf{k}_e, m)$ : On input a secret key  $\mathbf{k}_e$  for epoch  $e$  and a message  $m \in \mathcal{M}$ , it outputs a ciphertext  $\text{ct}_e$ .
- $\text{ct}_{e+1} \stackrel{\$}{\leftarrow} \text{UE.upd}(\Delta_{e+1}, \text{ct}_e)$ : On input a ciphertext  $\text{ct}_e$  from epoch  $e$  and the update token  $\Delta_{e+1}$ , it returns the updated ciphertext  $\text{ct}_{e+1}$ .
- $m'/\perp \leftarrow \text{UE.dec}(\mathbf{k}_e, \text{ct}_e)$ : On input a ciphertext  $\text{ct}_e$  and a secret key  $\mathbf{k}_e$  of some epoch  $e$ , it returns a message  $m'$  or  $\perp$ .

<p><u>Setup(<math>1^\lambda</math>):</u></p> $\mathbf{k}_0 \stackrel{\$}{\leftarrow} \text{UE.setup}(1^\lambda)$ $e := 0; \text{phase} := 0$ $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{K}, \mathcal{T}, \mathcal{C} \stackrel{\$}{\leftarrow} \emptyset$ <p><u><math>\mathcal{O}.\text{enc}(m)</math>:</u></p> $\text{ct} \stackrel{\$}{\leftarrow} \text{UE.enc}(\mathbf{k}_e, m)$ $\mathcal{L} := \mathcal{L} \cup \{(e, \text{ct})\}$ <p><b>return</b> ct</p> <p><u><math>\mathcal{O}.\text{next}</math>:</u></p> $e := e + 1$ $(\mathbf{k}_e, \Delta_e) \stackrel{\$}{\leftarrow} \text{UE.next}(\mathbf{k}_{e-1})$ <p><b>if</b> phase = 1 <b>then</b></p> $\tilde{\text{ct}}_e \stackrel{\$}{\leftarrow} \text{UE.upd}(\Delta_e, \tilde{\text{ct}}_{e-1})$ $\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup \{(e, \tilde{\text{ct}}_e)\}$ <p><u><math>\mathcal{O}.\text{upd}(\text{ct}_{e-1})</math>:</u></p> <p><b>if</b> <math>(e - 1, \text{ct}_{e-1}) \notin \mathcal{L}</math> <b>then</b></p> <p><b>return</b> <math>\perp</math></p> $\text{ct}_e \stackrel{\$}{\leftarrow} \text{UE.upd}(\Delta_e, \text{ct}_{e-1})$ $\mathcal{L} := \mathcal{L} \cup \{(e, \text{ct}_e)\}$ <p><b>return</b> <math>\text{ct}_e</math></p> <p><u><math>\mathcal{O}.\text{corr}(\text{inp}, \hat{e})</math>:</u></p> <p><b>if</b> <math>\hat{e} &gt; e</math> <b>then</b></p> <p><b>return</b> <math>\perp</math></p> <p><b>if</b> inp = key <b>then</b></p> $\mathcal{K} := \mathcal{K} \cup \{\hat{e}\}$ <p><b>return</b> <math>\mathbf{k}_{\hat{e}}</math></p> <p><b>if</b> inp = token <b>then</b></p> $\mathcal{T} := \mathcal{T} \cup \{\hat{e}\}$ <p><b>return</b> <math>\Delta_{\hat{e}}</math></p>	<p><u><math>\mathcal{O}.\text{chall-IND-ENC}(\bar{m}_0, \bar{m}_1)</math>:</u></p> <p><b>if</b> <math> \bar{m}_0  \neq  \bar{m}_1 </math> <b>then</b></p> <p><b>return</b> <math>\perp</math></p> <p>phase := 1; <math>\tilde{e} := e</math></p> $\tilde{\text{ct}}_e \stackrel{\$}{\leftarrow} \text{UE.enc}(\mathbf{k}_{\tilde{e}}, \bar{m}_b)$ $\mathcal{C} := \mathcal{C} \cup \{\tilde{e}\}$ $\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup \{(\tilde{e}, \tilde{\text{ct}}_e)\}$ <p><b>return</b> <math>\tilde{\text{ct}}_e</math></p> <p><u><math>\mathcal{O}.\text{chall-IND-UPD}(\bar{\text{ct}}_0, \bar{\text{ct}}_1)</math>:</u></p> <p><b>if</b> <math>(e - 1, \bar{\text{ct}}_0) \notin \mathcal{L}</math> or <math>(e - 1, \bar{\text{ct}}_1) \notin \mathcal{L}</math> or <math> \bar{\text{ct}}_0  \neq  \bar{\text{ct}}_1 </math> <b>then</b></p> <p><b>return</b> <math>\perp</math></p> <p>phase := 1; <math>\tilde{e} := e</math></p> $\tilde{\text{ct}}_e \stackrel{\$}{\leftarrow} \text{UE.upd}(\Delta_{\tilde{e}}, \bar{\text{ct}}_b)$ $\mathcal{C} := \mathcal{C} \cup \{\tilde{e}\}$ $\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup \{(\tilde{e}, \tilde{\text{ct}}_e)\}$ <p><b>return</b> <math>\tilde{\text{ct}}_e</math></p> <p><u><math>\mathcal{O}.\text{chall-IND-UE}(\bar{m}, \bar{\text{ct}})</math>:</u></p> <p><b>if</b> <math>(e - 1, \bar{\text{ct}}) \notin \mathcal{L}</math> <b>then</b></p> <p><b>return</b> <math>\perp</math></p> <p>phase := 1; <math>\tilde{e} := e</math></p> <p><b>if</b> b = 0 <b>then</b></p> $\tilde{\text{ct}}_e \stackrel{\$}{\leftarrow} \text{UE.enc}(\mathbf{k}_{\tilde{e}}, \bar{m})$ <p><b>else</b></p> $\tilde{\text{ct}}_e \stackrel{\$}{\leftarrow} \text{UE.upd}(\Delta_{\tilde{e}}, \bar{\text{ct}})$ $\mathcal{C} := \mathcal{C} \cup \{\tilde{e}\}$ $\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup \{(\tilde{e}, \tilde{\text{ct}}_e)\}$ <p><b>return</b> <math>\tilde{\text{ct}}_e</math></p> <p><u><math>\mathcal{O}.\text{upd}\tilde{\mathcal{C}}</math>:</u></p> <p><b>if</b> phase = 0 <b>then</b></p> <p><b>return</b> <math>\perp</math></p> $\mathcal{C} := \mathcal{C} \cup \{e\}$ <p><b>return</b> <math>\tilde{\text{ct}}_e</math></p>
--	--

Fig. 4. Oracles in security games for updatable encryption.

We stress that  $\text{UE.next}$  generates a new key along with an update token, which follows from the definition in the work of Lehmann and Tackmann [LT18]. In our constructions, the update token  $\Delta_{e+1}$  can also be generated from  $\mathbf{k}_e$  and  $\mathbf{k}_{e+1}$ .

**Definition 6 (Correctness).** Let  $\text{UE} = (\text{UE.setup}, \text{UE.next}, \text{UE.enc}, \text{UE.upd}, \text{UE.dec})$  be an updatable encryption scheme. We say UE is correct if for any  $m \in \mathcal{M}$ , any  $\mathbf{k}_0 \stackrel{\$}{\leftarrow} \text{UE.setup}(1^\lambda)$ , any sequence of  $(\mathbf{k}_1, \Delta_1), \dots, (\mathbf{k}_e, \Delta_e)$  gener-

ated as  $(\mathbf{k}_i, \Delta_i) \stackrel{\$}{\leftarrow} \text{UE.next}(\mathbf{k}_{i-1})$  for all  $i \in [e]$ , and for any  $0 \leq \hat{e} \leq e$ , let  $\text{ct}_{\hat{e}} \stackrel{\$}{\leftarrow} \text{UE.enc}(\mathbf{k}_{\hat{e}}, m)$  and  $\text{ct}_j \stackrel{\$}{\leftarrow} \text{UE.upd}(\Delta_j, \text{ct}_{j-1})$  for all  $j = \hat{e} + 1, \dots, e$ , then  $\text{UE.dec}(\mathbf{k}_e, \text{ct}_e) = m$ .

**Confidentiality.** The adversary  $\mathcal{A}$  has access to the oracles defined in Fig. 4. We follow the bookkeeping techniques of [LT18, KLR19, BDGJ20, Jia20], using the following sets to keep track of the generated and updated ciphertexts, and the epochs in which the adversary corrupted a key or a token, or learned a version of the challenge-ciphertext.

- $\mathcal{L}$ : Set of non-challenge ciphertexts  $(e, \text{ct}_e)$  produced by calls to the  $\mathcal{O}.enc$  or  $\mathcal{O}.upd$  oracle.  $\mathcal{O}.upd$  only updates ciphertexts obtained in  $\mathcal{L}$ .
- $\tilde{\mathcal{L}}$ : Set of updated versions of the challenge ciphertexts  $(e, \tilde{\text{ct}}_e)$ .  $\tilde{\mathcal{L}}$  is initiated with the challenge ciphertext  $(\tilde{e}, \tilde{\text{ct}}_{\tilde{e}})$ . Any call to the  $\mathcal{O}.next$  oracle automatically updates the challenge ciphertext to the new epoch, which the adversary can fetch via a call to  $\mathcal{O}.upd\tilde{\mathcal{C}}$ .
- $\mathcal{K}$ : Set of epochs  $e$  in which the adversary corrupted the secret key  $\mathbf{k}_e$  (from  $\mathcal{O}.corr$ ).
- $\mathcal{T}$ : Set of epochs  $e$  in which the adversary corrupted the update token  $\Delta_e$  (from  $\mathcal{O}.corr$ ).
- $\mathcal{C}$ : Set of epochs  $e$  in which the adversary learned a version of the challenge ciphertext (from  $\mathcal{O}.chall$  or  $\mathcal{O}.upd\tilde{\mathcal{C}}$ ).

We further define the epoch identification sets  $\mathcal{C}^*, \mathcal{K}^*, \mathcal{T}^*$  as the extended sets of  $\mathcal{C}, \mathcal{K}, \mathcal{T}$  in which the adversary learned or inferred information. We focus on *no-directional* key updates and *uni-directional* ciphertext updates.

$$\begin{aligned} \mathcal{K}^* &:= \mathcal{K} \\ \mathcal{T}^* &:= \{e \in \{0, \dots, e_{\text{end}}\} \mid (e \in \mathcal{T}) \vee (e - 1 \in \mathcal{K}^* \wedge e \in \mathcal{K}^*)\} \\ \mathcal{C}^* &:= \{e \in \{0, \dots, e_{\text{end}}\} \mid \text{ChallEq}(e) = \text{true}\} \\ &\text{where } \text{true} \leftarrow \text{ChallEq}(e) \iff (e \in \mathcal{C}) \vee (\text{ChallEq}(e - 1) \wedge e \in \mathcal{T}^*) \end{aligned}$$

*Remark 1.* The constructions we present later will in fact permit *backward-leak key updates*. At first glance the *backward-leak key updates* notion proposed by Nishimaki [Nis21] is seemingly weaker than *no-directionality key updates*. However, as mentioned in the introduction, this notion is essentially equivalent to no-directional key updates because backward-leak derivation of  $\mathbf{k}_{e-1}$  does not increase the adversary’s advantage in breaking the scheme. In particular, if the adversary obtains a challenge ciphertext  $\tilde{\text{ct}}_{e-1}$  and corrupts  $\Delta_e$  and  $\mathbf{k}_e$ , then it does *not* matter if the adversary can derive  $\mathbf{k}_{e-1}$  or not, as it can always update the ciphertext to  $\tilde{\text{ct}}_e$  and decrypt it using  $\mathbf{k}_e$ .

**Definition 7 (IND-ENC, IND-UPD, IND-UE security).** Let  $\text{UE} = (\text{UE.setup}, \text{UE.next}, \text{UE.enc}, \text{UE.upd}, \text{UE.dec})$  be an updatable encryption scheme. We say UE



is notion-secure for notion  $\in \{\text{IND-ENC}, \text{IND-UPD}, \text{IND-UE}\}$  if for all PPT adversary  $\mathcal{A}$  it holds that

$$\left| \Pr \left[ \text{Exp}_{\mathcal{A}, \text{UE}}^{\text{notion}}(1^\lambda) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$$

for some negligible function  $\text{negl}(\cdot)$ .

**Experiment**  $\text{Exp}_{\mathcal{A}, \text{UE}}^{\text{notion}}(1^\lambda)$  :

Run  $\text{Setup}(1^\lambda)$   
 (state,  $\text{Chall}_0, \text{Chall}_1$ )  $\stackrel{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}.\text{enc}, \mathcal{O}.\text{next}, \mathcal{O}.\text{upd}, \mathcal{O}.\text{corr}}(1^\lambda)$   
 $b \stackrel{\$}{\leftarrow} \{0, 1\}$   
 $\tilde{\text{ct}} \stackrel{\$}{\leftarrow} \mathcal{O}.\text{chall-notion}(\text{Chall}_0, \text{Chall}_1)$   
 Proceed only if  $\tilde{\text{ct}} \neq \perp$   
 $b' \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}.\text{enc}, \mathcal{O}.\text{next}, \mathcal{O}.\text{upd}, \mathcal{O}.\text{corr}, \mathcal{O}.\text{upd}\tilde{\text{C}}}(\text{state}, \tilde{\text{ct}})$   
**return** 1 if  $b = b'$  and  $\mathcal{C}^* \cap \mathcal{K}^* = \emptyset$

### 3.2 IND-ENC Secure Unidirectional UE

We begin by showing that any KPHE scheme with  $2n$ -bit secret keys that satisfies distributional semantic security with respect to the distribution  $\mathcal{U}_n$ , as well as public key and ciphertext blinding as described in Sect. 2 implies an IND-ENC secure unidirectional UE scheme.

**Construction.** Given a KHPE scheme of the form

$$\text{KPHE} = (\text{KPHE.Setup}, \text{KPHE.SKGen}, \text{KPHE.PKGen}, \text{KPHE.Enc}, \text{KPHE.Dec}, \text{KPHE.Eval}),$$

with  $2n$ -bit secret keys, we construct a unidirectional UE scheme

$$\text{UE} = (\text{UE.setup}, \text{UE.next}, \text{UE.enc}, \text{UE.upd}, \text{UE.dec}),$$

with message space  $\mathcal{M} = \{0, 1\}^{2n}$  as follows:

- $\text{UE.setup}(1^\lambda)$ : Generate  $\text{pp} \stackrel{\$}{\leftarrow} \text{KPHE.Setup}(1^\lambda)$ ,  $\text{sk}_0 \stackrel{\$}{\leftarrow} \text{KPHE.SKGen}(\text{pp}, \mathcal{U}_n)$ , and output
 
$$\mathbf{k}_0 = (\text{pp}, \text{sk}_0).$$
- $\text{UE.next}(\mathbf{k}_e)$ : Parse  $\mathbf{k}_e = (\text{pp}, \text{sk}_e)$ . Generate  $\text{sk}_{e+1} \stackrel{\$}{\leftarrow} \text{KPHE.SKGen}(\text{pp}, \mathcal{U}_n)$  and  $\text{pk}_{e+1} \stackrel{\$}{\leftarrow} \text{KPHE.PKGen}(\text{pp}, \text{sk}_{e+1})$ . Output
 
$$\mathbf{k}_{e+1} = (\text{pp}, \text{sk}_{e+1}), \quad \Delta_{e+1} = (\text{pk}_{e+1}, \text{KPHE.Enc}(\text{pk}_{e+1}, \text{sk}_e)).$$
- $\text{UE.enc}(\mathbf{k}_e, m)$ : Parse  $\mathbf{k}_e = (\text{pp}, \text{sk}_e)$ . Generate  $\text{pk}_e \stackrel{\$}{\leftarrow} \text{KPHE.PKGen}(\text{pp}, \text{sk}_e)$  and compute  $\text{ct}_{x_e} \stackrel{\$}{\leftarrow} \text{KPHE.Enc}(\text{pk}_e, m)$ . Output
 
$$\text{ct}_e = (0(\text{pk}_e, \text{ct}_{x_e})).$$

- **UE.upd**( $\Delta_{e+1}, \text{ct}_e$ ): Parse the update token and the ciphertext as

$$\Delta_{e+1} = (\text{pk}_\Delta, \text{ctx}_\Delta), \quad \text{ct}_e = (t, (\overline{\text{pk}}_{e-t}, \overline{\text{ctx}}_{e-t}), \dots, (\overline{\text{pk}}_{e-1}, \overline{\text{ctx}}_{e-1}), (\text{pk}_e, \text{ctx}_e))$$

Sample a uniform random permutation  $\pi : [2n] \rightarrow [2n]$ . Also, let  $\pi_{\text{id}} : [2n] \rightarrow [2n]$  denote the *identity* permutation. Compute

$$(\overline{\text{pk}}_e, \overline{\text{ctx}}_e) \stackrel{\$}{\leftarrow} \text{KPHE.Eval}(\text{pk}_e, \text{ctx}_e, \pi, \pi_{\text{id}}), \quad (\text{pk}_{e+1}, \text{ctx}_{e+1}) \stackrel{\$}{\leftarrow} \text{KPHE.Eval}(\text{pk}_\Delta, \text{ctx}_\Delta, \pi_{\text{id}}, \pi).$$

and output the updated ciphertext as:

$$\text{ct}_{e+1} = (t + 1, (\overline{\text{pk}}_{e-t}, \overline{\text{ctx}}_{e-t}), \dots, (\overline{\text{pk}}_e, \overline{\text{ctx}}_e), (\text{pk}_{e+1}, \text{ctx}_{e+1})).$$

- **UE.dec**( $\mathbf{k}_e, \text{ct}_e$ ): Parse  $\mathbf{k}_e = (\text{pp}, \text{sk}_e)$  and the ciphertext as

$$\text{ct}_e = (t, (\overline{\text{pk}}_{e-t}, \overline{\text{ctx}}_{e-t}), \dots, (\overline{\text{pk}}_{e-1}, \overline{\text{ctx}}_{e-1}), (\text{pk}_e, \text{ctx}_e)).$$

If  $t = 0$ , then output  $\mathbf{m} \leftarrow \text{KPHE.Dec}(\text{sk}_e, \text{ctx}_e)$ .

Otherwise, compute  $\overline{\text{sk}}_{e-1} \leftarrow \text{KPHE.Dec}(\text{sk}_e, \text{ctx}_e)$ . Then for each  $j$  from  $(e-1)$  downto  $(e-t+1)$ , compute

$$\overline{\text{sk}}_{j-1} \leftarrow \text{KPHE.Dec}(\overline{\text{sk}}_j, \overline{\text{ctx}}_j).$$

Finally, output the message  $\mathbf{m} \leftarrow \text{KPHE.Dec}(\overline{\text{sk}}_{e-t}, \overline{\text{ctx}}_{e-t})$ .

**Correctness.** We first prove the correctness of the UE scheme. For any  $\mathbf{m} \in \mathcal{M}$ , any  $\mathbf{k}_0 \leftarrow \text{UE.setup}(1^\lambda)$ , any sequence of  $(\mathbf{k}_1, \Delta_1), \dots, (\mathbf{k}_e, \Delta_e)$  generated as  $(\mathbf{k}_i, \Delta_i) \leftarrow \text{UE.next}(\mathbf{k}_{i-1})$  for all  $i \in [e]$ , let  $\text{ct}_0 \leftarrow \text{UE.enc}(\mathbf{k}_0, \mathbf{m})$  and  $\text{ct}_i \leftarrow \text{UE.upd}(\Delta_i, \text{ct}_{i-1})$  for all  $j \in [e]$ , then the final ciphertext is of the form  $\text{ct}_e = (e, (\overline{\text{pk}}_0, \overline{\text{ctx}}_0), \dots, (\overline{\text{pk}}_{e-1}, \overline{\text{ctx}}_{e-1}), (\text{pk}_e, \text{ctx}_e))$ . All the secret keys are of the form  $\mathbf{k}_0 = (\text{pp}, \text{sk}_0), \dots, \mathbf{k}_e = (\text{pp}, \text{sk}_e)$ . Let  $\pi_j$  be the random permutation sampled in  $\text{UE.upd}(\Delta_{j+1}, \text{ct}_j)$  and let  $\overline{\text{sk}}_j = \pi_j(\text{sk}_j)$  for all  $j = 0, 1, \dots, e-1$ . We can prove by induction that  $\text{KPHE.Dec}(\overline{\text{sk}}_0, \overline{\text{ctx}}_0) = \mathbf{m}$ ,  $\text{KPHE.Dec}(\overline{\text{sk}}_1, \overline{\text{ctx}}_1) = \overline{\text{sk}}_0, \dots, \text{KPHE.Dec}(\overline{\text{sk}}_{e-1}, \overline{\text{ctx}}_{e-1}) = \overline{\text{sk}}_{e-2}$ ,  $\text{KPHE.Dec}(\text{sk}_e, \text{ctx}_e) = \overline{\text{sk}}_{e-1}$ . Therefore,  $\text{UE.dec}(\mathbf{k}_e, \text{ct}_e)$  outputs  $\mathbf{m}$ . This argument is for any ciphertext starting from epoch 0. The same argument holds for any ciphertext starting from any epoch  $\hat{e}$  where  $0 \leq \hat{e} \leq e$ .

**Confidentiality.** Next we prove the IND-ENC security of our UE scheme. More formally, we state and prove the following theorem:

**Theorem 3 (IND-ENC Security).** *Assuming that KPHE satisfies distributional security with respect to the distribution  $\mathcal{U}_n$ , as well as public key and ciphertext blinding as described in Sect. 2, the above UE construction is an IND-ENC secure unidirectional UE scheme.*

*Proof.* The proof proceeds via a hybrid argument.

$\text{Hyb}_0$  The challenger plays the real game with the adversary.

Hyb<sub>1</sub> Same as Hyb<sub>0</sub> but for  $\text{UE.upd}(\Delta_e, \text{ct}_{e-1})$  in  $\mathcal{O.upd}$  and  $\text{UE.upd}(\Delta_e, \tilde{\text{ct}}_{e-1})$  in  $\mathcal{O.next}$ , do the following:

- Let  $\mathbf{k}_{e-1} = (\text{pp}, \text{sk}_{e-1})$  and  $\mathbf{k}_e = (\text{pp}, \text{sk}_e)$ .
- Parse the ciphertext  $\text{ct}_{e-1}$  or  $\tilde{\text{ct}}_{e-1}$  as

$$(t, (\overline{\text{pk}}_{e-1-t}, \overline{\text{ct}}_{e-1-t}), \dots, (\overline{\text{pk}}_{e-2}, \overline{\text{ct}}_{e-2}), (\text{pk}_{e-1}, \text{ct}_{e-1})),$$

where  $\text{ct}_{e-1} = \text{KPHE.Enc}(\text{pk}_{e-1}, x)$ . Note that if  $t = 0$ , then  $x = m$  for some message, otherwise  $x = \overline{\text{sk}}_{e-2}$  that is the KPHE secret key corresponding to  $\overline{\text{pk}}_{e-2}$ .

- Sample a uniform random permutation  $\pi : [2n] \rightarrow [2n]$ , let  $\overline{\text{sk}}_{e-1} = \pi(\text{sk}_{e-1})$ , and sample  $\overline{\text{pk}}_{e-1} \stackrel{\$}{\leftarrow} \text{KPHE.PKGen}(\text{pp}, \overline{\text{sk}}_{e-1})$ . Compute  $\overline{\text{ct}}_{e-1} \stackrel{\$}{\leftarrow} \text{KPHE.Enc}(\overline{\text{pk}}_{e-1}, x)$ .
- Sample  $\text{pk}_e \stackrel{\$}{\leftarrow} \text{KPHE.PKGen}(\text{pp}, \text{sk}_e)$  and compute  $\text{ct}_e \stackrel{\$}{\leftarrow} \text{KPHE.Enc}(\text{pk}_e, \overline{\text{sk}}_{e-1})$ .
- Let  $\text{ct}_e$  or  $\tilde{\text{ct}}_e$  be

$$(t+1, (\overline{\text{pk}}_{e-1-t}, \overline{\text{ct}}_{e-1-t}), \dots, (\overline{\text{pk}}_{e-1}, \overline{\text{ct}}_{e-1}), (\text{pk}_e, \text{ct}_e)).$$

We prove in Lemma 2 that this hybrid is computationally indistinguishable from Hyb<sub>0</sub> to any PPT adversary by the blinding property of KPHE.

Hyb<sub>2</sub> Same as Hyb<sub>1</sub> but for  $\text{UE.upd}(\Delta_e, \text{ct}_{e-1})$  in  $\mathcal{O.upd}$  and  $\text{UE.upd}(\Delta_e, \tilde{\text{ct}}_{e-1})$  in  $\mathcal{O.next}$ , instead of letting  $\overline{\text{sk}}_{e-1} = \pi(\text{sk}_{e-1})$ , sample  $\overline{\text{sk}}_{e-1}$  from the distribution  $\mathcal{U}_n$ . This hybrid is statistically identical to Hyb<sub>1</sub>.

Hyb<sub>3</sub> Let  $\tilde{e}$  be the challenge epoch, and let  $\bar{e}$  be the last epoch where the adversary corrupts continuous update tokens from  $\tilde{e}$ , namely the adversary corrupts  $\Delta_{\tilde{e}+1}, \Delta_{\tilde{e}+2}, \dots, \Delta_{\bar{e}}$  but not  $\Delta_{\bar{e}+1}$ . This hybrid is the same as Hyb<sub>2</sub> except that the challenger guesses  $\tilde{e}^*$  and  $\bar{e}^*$  at the beginning of the game and aborts the game if guessing incorrectly. Let  $E$  be the upper bound on the number of epochs during the game. If the challenger does not abort, then this hybrid is identical to Hyb<sub>2</sub>, which happens with probability at least  $\frac{1}{E^2}$ . In the remaining hybrids, we assume for simplicity that the challenger guesses  $\tilde{e}$  and  $\bar{e}$  correctly.

Hyb<sub>4</sub> Same as Hyb<sub>3</sub> except that for each  $\mathbf{k}_e = (\text{pp}, \text{sk}_e)$ , generate a single public key  $\widehat{\text{pk}}_e \stackrel{\$}{\leftarrow} \text{KPHE.PKGen}(\text{pp}, \text{sk}_e)$ . Then whenever  $\text{KPHE.Enc}(\text{pk}_e, x)$  is computed for a freshly generated  $\text{pk}_e$  and some  $x$ , compute it as  $\text{KPHE.Eval}(\widehat{\text{pk}}_e, \text{KPHE.Enc}(\widehat{\text{pk}}_e, x), \pi_{\text{id}}, \pi_{\text{id}})$ . That is, instead of generating a fresh  $\text{pk}_e$  from  $\text{sk}_e$  every time, use the same  $\widehat{\text{pk}}_e$  to encrypt  $x$  and use then  $\text{KPHE.Eval}$  to re-randomize it.

This hybrid is computationally indistinguishable from Hyb<sub>3</sub> by the blinding property of KPHE. We omit the detailed reduction here, but it is similar to the reduction in the proof of Lemma 2.

Hyb<sub>5</sub> Same as Hyb<sub>4</sub> except that for all  $\tilde{e} + 1 \leq e \leq \bar{e}$ ,  $\text{UE.next}(\mathbf{k}_{e-1})$  is computed as follows. Generate  $\text{sk}_e \stackrel{\$}{\leftarrow} \text{KPHE.SKGen}(\text{pp}, \mathcal{U}_n)$  and let  $\widehat{\text{pk}}_e \stackrel{\$}{\leftarrow}$

$\text{KPHE.PKGen}(\text{pp}, \text{sk}_e)$  be the single public key for  $\mathbf{k}_e$  (that will be used for every  $\text{KPHE.Enc}$ ). Output

$$\mathbf{k}_e = (\text{pp}, \text{sk}_e), \quad \Delta_e = (\widehat{\text{pk}}_e, \text{KPHE.Enc}(\widehat{\text{pk}}_e, 0^{2n})).$$

We prove in Lemma 3 that this hybrid is computationally indistinguishable from  $\text{Hyb}_4$  to any PPT adversary based on the distributional semantic security of KPHE.

$\text{Hyb}_6$  Same as  $\text{Hyb}_5$  except that for each  $\mathbf{k}_e = (\text{pp}, \text{sk}_e)$ , generate a single public key  $\widehat{\text{pk}}_e \xleftarrow{\$} \text{KPHE.PKGen}(\text{pp}, \text{sk}_e)$  and use  $\text{KPHE.Eval}(\widehat{\text{pk}}_e, \text{KPHE.Enc}(\widehat{\text{pk}}_e, \cdot), \pi_{\text{id}}, \pi_{\text{id}})$  for all the computation of  $\text{KPHE.Enc}(\mathbf{k}_e, \cdot)$  (including the computation of  $\Delta_e$ ). The only exception is the challenge ciphertext  $\tilde{\text{ct}}_e$ , which is computed using  $\widehat{\text{pk}}_e$  without re-randomization, namely

$$\tilde{\text{ct}}_e = (0, (\widehat{\text{pk}}_e, \text{KPHE.Enc}(\widehat{\text{pk}}_e, \bar{\mathbf{m}}_b))).$$

This hybrid is computationally indistinguishable from  $\text{Hyb}_5$  by the blinding property of KPHE. We omit the detailed reduction here, but it is similar to the reduction in the proof of Lemma 2.

Finally, we argue that in the final hybrid  $\text{Hyb}_6$ , any PPT adversary cannot distinguish an encryption of  $\bar{\mathbf{m}}_0$  or  $\bar{\mathbf{m}}_1$  in the challenge epoch  $\tilde{e}$ , which relies on the distributional semantic security of KPHE, which will conclude our proof.

Assume for the purpose of contradiction that there exists a PPT adversary  $\mathcal{A}$  that can distinguish an encryption of  $\bar{\mathbf{m}}_0$  or  $\bar{\mathbf{m}}_1$  in the challenge epoch. Then we construct a PPT adversary  $\mathcal{B}$  that breaks the distributional semantic security of KPHE. The adversary  $\mathcal{B}$  first receives  $(\text{pp}, \text{pk})$  from the challenger in the semantic security game. Then  $\mathcal{B}$  plays the UE game with  $\mathcal{A}$  as a challenger in  $\text{Hyb}_6$ .  $\mathcal{B}$  uses  $\text{pp}$  to generate UE keys and update tokens as in  $\text{Hyb}_6$  except that for epoch  $\tilde{e}$ , the UE key  $\mathbf{k}_{\tilde{e}}$  is unknown. When  $\mathcal{B}$  receives the challenge messages  $(\bar{\mathbf{m}}_0, \bar{\mathbf{m}}_1)$  from  $\mathcal{A}$  in the UE game, it forwards the two messages to the KPHE challenger and gets back  $\text{ctx}$ , and then responds to  $\mathcal{A}$  with  $\text{ct} = (0, (\text{pp}, \text{ctx}))$ . Note that  $\mathcal{B}$  doesn't need to know  $\mathbf{k}_{\tilde{e}}$  because it is never used. In particular,  $\mathcal{B}$  can use  $\text{pk}$  to compute all the  $\text{UE.enc}(\mathbf{k}_{\tilde{e}}, \cdot)$ . Finally,  $\mathcal{B}$  outputs whatever  $\mathcal{A}$  outputs.

If  $\mathcal{A}$  can distinguish between encryptions of  $\bar{\mathbf{m}}_0$  and  $\bar{\mathbf{m}}_1$  with non-negligible probability, then  $\mathcal{B}$  can break the distributional semantic security of KPHE with non-negligible probability, which leads to contradiction.

**Lemma 2.**  $\text{Hyb}_0 \stackrel{c}{\approx} \text{Hyb}_1$  in the proof of Theorem 3.

**Lemma 3.**  $\text{Hyb}_4 \stackrel{c}{\approx} \text{Hyb}_5$  in the proof of Theorem 3.

We defer the formal proofs of Lemmas 2 and 3 to the full version of our paper [MPW22]. These proofs complete the overall proof of Theorem 3.  $\square$

*Remark 2.* In our construction one can derive  $\mathbf{k}_{e-1}$  from  $\Delta_e$  and  $\mathbf{k}_e$ . It is for this reason that our construction permits *backward-leak unidirectional key updates* proposed by Nishimaki [Nis21] where secret keys can be derived in the backward direction but not forward direction. However, as discussed earlier, this notion is essentially equivalent to no-directional key updates (the optimal case) and has no bearing on our security analysis.

### 3.3 Post-Compromise Secure Unidirectional UE

In this section, we show that any KPHE scheme with  $2n$ -bit secret keys that satisfies distributional security with respect to the distribution  $\mathcal{U}_n$ , as well as public key and ciphertext blinding as described in Sect. 2 implies a post-compromise secure unidirectional UE scheme.

#### 3.3.1 IND-UPD Secure Unidirectional UE

We first show how to construct an UE scheme that satisfies the IND-UPD security definition as proposed in [LT18]. Given a KPHE scheme of the form

$$\text{KPHE} = (\text{KPHE.Setup}, \text{KPHE.KeyGen}, \text{KPHE.Enc}, \text{KPHE.Dec}, \text{KPHE.TransPK}, \text{KPHE.Eval}),$$

with  $2n$ -bit secret keys, we construct a unidirectional UE scheme

$$\text{UE} = (\text{UE.setup}, \text{UE.next}, \text{UE.enc}, \text{UE.upd}, \text{UE.dec}),$$

that only differs from the IND-ENC construction in  $\text{UE.upd}$ :

- $\text{UE.upd}(\Delta_{e+1}, \text{ct}_e)$ : Parse the update token and the ciphertext as

$$\Delta_{e+1} = (\text{pk}_\Delta, \text{ctx}_\Delta), \quad \text{ct}_e = (t, (\overline{\text{pk}}_{e-t}, \overline{\text{ctx}}_{e-t}), \dots, (\overline{\text{pk}}_{e-1}, \overline{\text{ctx}}_{e-1}), (\text{pk}_e, \text{ctx}_e))$$

Sample  $(t + 1)$  uniform random permutations  $\pi_{e-t}, \dots, \pi_e : [2n] \rightarrow [2n]$ . Also, let  $\pi_{\text{id}} : [2n] \rightarrow [2n]$  denote the identity permutation. For each  $i \in \{e - t + 1, \dots, e - 1\}$ , compute

$$(\widetilde{\text{pk}}_i, \widetilde{\text{ctx}}_i) \stackrel{\$}{\leftarrow} \text{KPHE.Eval}(\overline{\text{pk}}_i, \overline{\text{ctx}}_i, \pi_i, \pi_{i-1}).$$

Additionally, compute

$$(\widetilde{\text{pk}}_{e-t}, \widetilde{\text{ctx}}_{e-t}) \stackrel{\$}{\leftarrow} \text{KPHE.Eval}(\overline{\text{pk}}_{e-t}, \overline{\text{ctx}}_{e-t}, \pi_{e-t}, \pi_{\text{id}}),$$

$$(\overline{\text{pk}}_e, \overline{\text{ctx}}_e) \stackrel{\$}{\leftarrow} \text{KPHE.Eval}(\text{pk}_e, \text{ctx}_e, \pi_e, \pi_{e-1}),$$

$$(\text{pk}_{e+1}, \text{ctx}_{e+1}) \stackrel{\$}{\leftarrow} \text{KPHE.Eval}(\text{pk}_\Delta, \text{ctx}_\Delta, \pi_{\text{id}}, \pi_e).$$

Output the updated ciphertext as:

$$\text{ct}_{e+1} = (t + 1, (\widetilde{\text{pk}}_{e-t}, \widetilde{\text{ctx}}_{e-t}), \dots, (\widetilde{\text{pk}}_{e-1}, \widetilde{\text{ctx}}_{e-1}), (\overline{\text{pk}}_e, \overline{\text{ctx}}_e), (\text{pk}_{e+1}, \text{ctx}_{e+1})).$$

**Correctness.** We first prove the correctness of the UE scheme. For any  $m \in \mathcal{M}$ , any  $\mathbf{k}_0 \leftarrow \text{UE.setup}(1^\lambda)$ , any sequence of  $(\mathbf{k}_1, \Delta_1), \dots, (\mathbf{k}_e, \Delta_e)$  generated as  $(\mathbf{k}_i, \Delta_i) \leftarrow \text{UE.next}(\mathbf{k}_{i-1})$  for all  $i \in [e]$ , let  $\text{ct}_0 \leftarrow \text{UE.enc}(\mathbf{k}_0, m)$  and  $\text{ct}_i \leftarrow \text{UE.upd}(\Delta_i, \text{ct}_{i-1})$  for all  $j \in [e]$ , then the final ciphertext is of the form  $\text{ct}_e = (e, (\overline{\text{pk}}_0, \overline{\text{ct}}_{x_0}), \dots, (\overline{\text{pk}}_{e-1}, \overline{\text{ct}}_{x_{e-1}}), (\text{pk}_e, \text{ct}_x_e))$ . All the UE secret keys are of the form  $\mathbf{k}_0 = (\text{pp}, \text{sk}_0), \dots, \mathbf{k}_e = (\text{pp}, \text{sk}_e)$ . We can prove by induction that there exist permutations  $\pi_0, \pi_1, \dots, \pi_{e-1} : [2n] \rightarrow [2n]$  such that  $\overline{\text{sk}}_i = \pi_i(\text{sk}_i)$  for all  $i = 0, 1, \dots, e-1$ , and that  $\text{KPHE.Dec}(\overline{\text{sk}}_0, \overline{\text{ct}}_{x_0}) = m$ ,  $\text{KPHE.Dec}(\overline{\text{sk}}_1, \overline{\text{ct}}_{x_1}) = \overline{\text{sk}}_0, \dots, \text{KPHE.Dec}(\overline{\text{sk}}_{e-1}, \overline{\text{ct}}_{x_{e-1}}) = \overline{\text{sk}}_{e-2}$ ,  $\text{KPHE.Dec}(\text{sk}_e, \text{ct}_x_e) = \text{sk}_{e-1}$ . Therefore,  $\text{UE.dec}(\mathbf{k}_e, \text{ct}_e)$  outputs  $m$ . This argument is for any ciphertext starting from epoch 0. The same argument holds for any ciphertext starting from any epoch  $\hat{e}$  where  $0 \leq \hat{e} \leq e$ .

**Confidentiality.** Next we prove the IND-UPD security the UE scheme. More formally, we state and prove the following theorem (the proof is provided in the full version of our paper [MPW22]):

**Theorem 4 (IND-UPD Security).** *Assuming that KPHE satisfies distributional security with respect to the distribution  $\mathcal{U}_n$ , as well as public key and ciphertext blinding as described in Sect. 2, the above UE construction is an IND-UPD secure unidirectional UE scheme.*

### 3.3.2 IND-UE Secure Unidirectional UE

The basic IND-UPD construction allows ciphertexts from the same epoch  $e$  to have different sizes. In particular, a freshly created ciphertext in epoch  $e$  can be trivially distinguished from a ciphertext that was created as an update of a ciphertext from epoch  $(e-1)$ . So it cannot satisfy the combined security definition of post-compromise security for UE due to Boyd et al. [BDGJ20].

In the full version of our paper [MPW22], we showcase a simple extension of the basic construction wherein we ensure that the size for any ciphertext in epoch  $e$  is the same, irrespective of whether it was freshly created, or created as an update of a ciphertext from epoch  $(e-1)$ . The overall construction remains exactly the same; the key alteration is in how we generate fresh ciphertexts. At a high level, a freshly created ciphertext in epoch  $e$  is made to look exactly like a ciphertext that has undergone  $e$  update operations. We do this by having  $e$  “dummy wrapper” layers over and above the core ciphertext generated by the basic construction. We defer the detailed construction and security proof to the full version of our paper [MPW22].

## 4 Unidirectional PRE from Circular-Secure KPHE

In this section, we show how to construct unidirectional PRE from any KPHE scheme that satisfies distributional circular security. We present the simpler construction of IND-HRA unidirectional PRE in Sect. 4.2. In the full version of our

paper [MPW22], we show how to augment it to achieve the stronger notion of strong post-compromise security (PCS) as introduced in a recent work by Davidson et al. [DDL19].

#### 4.1 Definition of Unidirectional PRE

**Definition 8 (Unidirectional Proxy Re-Encryption (PRE)).** A unidirectional PRE scheme is a tuple of PPT algorithms of the form

$$\text{PRE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{ReKeyGen}, \text{ReEnc}, \text{Dec}),$$

described as follows:

- $\text{pp} \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda)$ : On input the security parameter  $\lambda$ , the setup algorithm outputs some public parameters  $\text{pp}$  (these parameters are implicit to all other algorithms).
- $(\text{sk}, \text{pk}) \stackrel{\$}{\leftarrow} \text{KeyGen}(\text{pp})$ : On input the public parameters  $\text{pp}$ , the key-generation algorithm outputs a secret key-public key pair,  $(\text{sk}, \text{pk})$ .
- $\text{ct} \stackrel{\$}{\leftarrow} \text{Enc}(\text{pk}, \text{m})$ : On input a public key  $\text{pk}$  and a message  $\text{m}$ , the encryption algorithm outputs a ciphertext  $\text{ct}$ .
- $\text{rk}_{i,j} \stackrel{\$}{\leftarrow} \text{ReKeyGen}((\text{sk}_i, \text{pk}_i), \text{pk}_j)$ : The re-key generation algorithm returns a re-encryption key  $\text{rk}_{i,j}$  for translation of a ciphertext from a key-pair  $(\text{sk}_i, \text{pk}_i)$  to a key-pair  $(\text{sk}_j, \text{pk}_j)$ . It takes as input  $(\text{sk}_i, \text{pk}_i)$  and  $\text{pk}_j$ , and outputs the re-encryption key  $\text{rk}_{i,j}$ .<sup>1</sup>
- $\text{ct}_j \stackrel{\$}{\leftarrow} \text{ReEnc}(\text{rk}_{i,j}, \text{ct}_i)$ : On input a re-encryption key  $\text{rk}_{i,j}$  and a ciphertext  $\text{ct}_i$ , the re-encryption algorithm outputs an updated ciphertext  $\text{ct}_j$ .<sup>2</sup>
- $\text{m}/\perp \leftarrow \text{Dec}(\text{sk}, \text{ct})$ : On input a secret key  $\text{sk}$  and a ciphertext  $\text{ct}$ , the decryption algorithm outputs either a plaintext message or an error symbol.

**Definition 9 (Correctness).** A PRE scheme  $\text{PRE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{ReKeyGen}, \text{ReEnc}, \text{Dec})$  is said to be correct if for any  $\text{pp} \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda)$ , for any  $\ell \geq 0$ , for any  $(\ell + 1)$  key-pairs  $(\text{pk}_0, \text{sk}_0), \dots, (\text{pk}_\ell, \text{sk}_\ell) \stackrel{\$}{\leftarrow} \text{KeyGen}(\text{pp})$ , and for any plaintext message  $\text{m}$ , letting  $\text{ct}_0 \stackrel{\$}{\leftarrow} \text{Enc}(\text{pk}_0, \text{m})$ , and letting for each  $j \in [\ell]$

$$\text{rk}_j \stackrel{\$}{\leftarrow} \text{ReKeyGen}(\text{sk}_{j-1}, \text{pk}_{j-1}, \text{pk}_j), \quad \text{ct}_j \stackrel{\$}{\leftarrow} \text{ReEnc}(\text{rk}_j, \text{ct}_{j-1}),$$

we have  $\text{Dec}(\text{sk}_\ell, \text{ct}_\ell) = \text{m}$  (with all but negligible probability).

**Confidentiality.** We defer the confidentiality definitions to the full version of our paper [MPW22].

<sup>1</sup> In a bidirectional PRE scheme, the re-key generation algorithm additionally takes as input the destination secret key  $\text{sk}_j$ , i.e., it takes as input  $(\text{sk}_i, \text{pk}_i)$  and  $(\text{sk}_j, \text{pk}_j)$ , and outputs the re-encryption key  $\text{rk}_{i,j}$ .

<sup>2</sup> The re-encryption algorithm could be either deterministic or randomized; in this work, we assume throughout that the re-encryption algorithm is randomized.

### 4.2 HRA-Secure Unidirectional PRE

We show that any KPHE scheme with  $2n$ -bit secret keys and plaintext messages that satisfies: (a) distributional semantic and *circular* security with respect to the distribution  $\mathcal{U}_n$ , and (b) blinding, implies the existence of a multi-hop IND-HRA secure unidirectional PRE scheme.

**Construction.** Given a KHPE scheme of the form

$$\text{KPHE} = (\text{KPHE.Setup}, \text{KPHE.SKGen}, \text{KPHE.PKGen}, \text{KPHE.Enc}, \text{KPHE.Dec}, \text{KPHE.Eval}),$$

with  $2n$ -bit secret keys, we construct a unidirectional PRE scheme

$$\text{PRE} = (\text{PRE.Setup}, \text{PRE.KeyGen}, \text{PRE.Enc}, \text{PRE.ReKeyGen}, \text{PRE.ReEnc}, \text{PRE.Dec}),$$

with message space  $\mathcal{M} = \{0, 1\}^{2n}$  as follows:

- $\text{PRE.Setup}(1^\lambda)$ : Sample  $\text{pp} \xleftarrow{\$} \text{KPHE.Setup}(1^\lambda)$  and output  $\text{pp}$ .
- $\text{PRE.KeyGen}(\text{pp})$ : Sample and output  $(\text{pk}, \text{sk})$  where

$$\text{sk} \xleftarrow{\$} \text{KPHE.SKGen}(\text{pp}, \mathcal{U}_n), \quad \text{pk} \xleftarrow{\$} \text{KPHE.PKGen}(\text{pp}, \text{sk}).$$

- $\text{PRE.Enc}(\text{pk}, m)$ : Compute  $\text{ctx}_0 \xleftarrow{\$} \text{KPHE.Enc}(\text{pk}, m)$  and output
 
$$\text{ct} = (0, (\text{pk}, \text{ctx}_0)).$$
- $\text{PRE.ReKeyGen}(\text{sk}_i, \text{pk}_i, \text{pk}_j)$ : Output  $\text{rk}_{i,j} = (\text{pk}_j, \text{ctx}_\Delta)$ , where

$$\text{ctx}_\Delta \xleftarrow{\$} \text{KPHE.Enc}(\text{pk}_j, \text{sk}_i).$$

- $\text{PRE.ReEnc}(\text{rk}_{i,j}, \text{ct})$ : Parse the reencryption key and the ciphertext as

$$\text{rk}_{i,j} = (\text{pk}_j, \text{ctx}_\Delta), \quad \text{ct} = (t, (\overline{\text{pk}}_0, \overline{\text{ctx}}_0), \dots, (\overline{\text{pk}}_{t-1}, \overline{\text{ctx}}_{t-1}), (\widehat{\text{pk}}_t, \widehat{\text{ctx}}_t)),$$

for some  $t \geq 0$ . Sample a uniformly random permutation  $\pi : [2n] \rightarrow [2n]$ . Also, let  $\pi_{\text{id}} : [2n] \rightarrow [2n]$  denote the *identity* permutation. Compute

$$(\overline{\text{pk}}_t, \overline{\text{ctx}}_t) \xleftarrow{\$} \text{KPHE.Eval}(\widehat{\text{pk}}_t, \widehat{\text{ctx}}_t, \pi, \pi_{\text{id}}),$$

$$(\widehat{\text{pk}}_{t+1}, \widehat{\text{ctx}}_{t+1}) \xleftarrow{\$} \text{KPHE.Eval}(\text{pk}_j, \text{ctx}_\Delta, \pi_{\text{id}}, \pi),$$

and output the updated ciphertext as:

$$\text{ct}' = (t + 1, (\overline{\text{pk}}_0, \overline{\text{ctx}}_0), \dots, (\overline{\text{pk}}_t, \overline{\text{ctx}}_t), (\widehat{\text{pk}}_{t+1}, \widehat{\text{ctx}}_{t+1})).$$



–  $\text{PRE.Dec}(\text{sk}, \text{ct})$ : Parse the ciphertext as

$$\text{ct} = (t, (\overline{\text{pk}}_0, \overline{\text{ctx}}_0), \dots, (\overline{\text{pk}}_{t-1}, \overline{\text{ctx}}_{t-1}), (\widehat{\text{pk}}_t, \widehat{\text{ctx}}_t)),$$

for some  $t \geq 0$ . Compute  $\overline{\text{sk}}_{t-1} = \text{KPHE.Dec}(\text{sk}, \widehat{\text{ctx}}_t)$ . Next, compute the following for each  $\ell$  from  $(t-1)$  to 1 in decreasing order:

$$\overline{\text{sk}}_{\ell-1} = \text{KPHE.Dec}(\overline{\text{sk}}_{\ell}, \overline{\text{ctx}}_{\ell}).$$

Finally, output the message  $\text{m} = \text{KPHE.Dec}(\overline{\text{sk}}_0, \overline{\text{ctx}}_0)$ .

**Correctness.** We defer the detailed proof of correctness to the full version of our paper [MPW22]. At a high level, the correctness argument is very similar to that for our unidirectional UE scheme in Sect. 3.2.

**Theorem 5 (IND-HRA Security).** *Assuming that KPHE satisfies blinding and distributional semantic+circular security with respect to the distribution  $\mathcal{U}_n$ , PRE is a multi-hop IND-HRA secure unidirectional PRE scheme.*

We defer the detailed proofs of correctness and IND-HRA security to the full version of our paper [MPW22]. Also, see [MPW22] for our construction of IND-PCS secure unidirectional PRE from any circular-secure KPHE scheme.

**Acknowledgements.** The authors thank the anonymous reviewers of IACR PKC 2023 for their helpful comments and suggestions. The authors did part of the work while at VISA Research. P. Miao is supported in part by the NSF CNS Award 2055358, a 2020 DPI Science Team Seed Grant, and a Meta award.

## References

- [ACDT20] Alwen, J., Coretti, S., Dodis, Y., Tseleounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 248–277. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-56784-2\\_9](https://doi.org/10.1007/978-3-030-56784-2_9)
- [ACPS09] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03356-8\\_35](https://doi.org/10.1007/978-3-642-03356-8_35)
- [AFGH06] Ateniese, G., Kevin, F., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (2006)
- [AMP19] Alamati, N., Montgomery, H., Patranabis, S.: Symmetric primitives with structured secrets. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 650–679. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26948-7\\_23](https://doi.org/10.1007/978-3-030-26948-7_23)
- [BBB+12] Barker, E., Barker, W., Burr, W., Polk, W., Smid, M., Gallagher, P.D., Under Secretary For.: NIST Special Publication 800–57 Recommendation for Key Management - Part 1: General (2012)



- [BBS98] Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054122>
- [BDGJ20] Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 464–493. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-56784-2\\_16](https://doi.org/10.1007/978-3-030-56784-2_16)
- [BEKS20] Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 559–589. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_19](https://doi.org/10.1007/978-3-030-64840-4_19)
- [BF03] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. *SIAM J. Comput.* **32**(3), 586–615 (2003)
- [BGI+12] Barak, B., et al.: On the (im)possibility of obfuscating programs. *J. ACM* **59**(2), 6:1–6:48 (2012)
- [BHHO08] Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85174-5\\_7](https://doi.org/10.1007/978-3-540-85174-5_7)
- [BLMR13] Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40041-4\\_23](https://doi.org/10.1007/978-3-642-40041-4_23)
- [BRS02] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36492-7\\_6](https://doi.org/10.1007/3-540-36492-7_6)
- [CCL+14] Chandran, N., Chase, M., Liu, F.-H., Nishimaki, R., Xagawa, K.: Re-encryption, functional re-encryption, and multi-hop re-encryption: a framework for achieving obfuscation-based security and instantiations from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 95–112. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54631-0\\_6](https://doi.org/10.1007/978-3-642-54631-0_6)
- [CL01] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44987-6\\_7](https://doi.org/10.1007/3-540-44987-6_7)
- [Coh19] Cohen, A.: What about bob? the inadequacy of CPA security for proxy reencryption. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 287–316. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17259-6\\_10](https://doi.org/10.1007/978-3-030-17259-6_10)
- [DDL19] Davidson, A., Deo, A., Lee, E., Martin, K.: Strong post-compromise secure proxy re-encryption. In: Jang-Jaccard, J., Guo, F. (eds.) ACISP 2019. LNCS, vol. 11547, pp. 58–77. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-21548-4\\_4](https://doi.org/10.1007/978-3-030-21548-4_4)
- [DKW21] Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13044, pp. 254–285. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-90456-2\\_9](https://doi.org/10.1007/978-3-030-90456-2_9)

- [EPRS17] Everspaugh, A., Paterson, K., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 98–129. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_4](https://doi.org/10.1007/978-3-319-63697-9_4)
- [FKKP19] Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 317–346. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17259-6\\_11](https://doi.org/10.1007/978-3-030-17259-6_11)
- [FL17] Fan, X., Liu, F.-H.: Proxy re-encryption and re-signatures from lattices. Cryptology ePrint Archive, Report 2017/456, 2017. <https://eprint.iacr.org/2017/456>
- [Gam85] El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theor. **31**(4), 469–472 (1985)
- [Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In ACM STOC **2009**, 169–178 (2009)
- [GHV10] Gentry, C., Halevi, S., Vaikuntanathan, V.: *i*-Hop homomorphic encryption and rerandomizable Yao circuits. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 155–172. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14623-7\\_9](https://doi.org/10.1007/978-3-642-14623-7_9)
- [GP23] Galteland, Y.J., Pan, J.: Backward-leak uni-directional updatable encryption from public key encryption. In: PKC 2023 (to appear) 2023. <https://eprint.iacr.org/2022/324>
- [GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: ACM STOC 2008, pp. 197–206. ACM (2008)
- [ID03] Ivan, A.-A., Dodis, Y.: Proxy cryptography revisited. In NDSS 2003. The Internet Society (2003)
- [Jia20] Jiang, Y.: The direction of updatable encryption does not matter much. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 529–558. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_18](https://doi.org/10.1007/978-3-030-64840-4_18)
- [JMM19] Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 159–188. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17653-2\\_6](https://doi.org/10.1007/978-3-030-17653-2_6)
- [Kir14] Kirshanova, E.: Proxy re-encryption from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 77–94. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54631-0\\_5](https://doi.org/10.1007/978-3-642-54631-0_5)
- [KLR19] Kloof, Michael, Lehmann, Anja, Rupp, Andy: (R)CCA Secure Updatable Encryption with Integrity Protection. In: Ishai, Yuval, Rijmen, Vincent (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 68–99. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17653-2\\_3](https://doi.org/10.1007/978-3-030-17653-2_3)
- [LT18] Lehmann, Anja, Tackmann, Björn.: Updatable encryption with post-compromise security. In: Nielsen, Jesper Buus, Rijmen, Vincent (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 685–716. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_22](https://doi.org/10.1007/978-3-319-78372-7_22)
- [LV11] Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. IEEE Trans. Inf. Theor. **57**(3), 1786–1802 (2011)
- [MPW22] Miao, P., Patranabis, S., Watson, G.: Unidirectional updatable encryption and proxy re-encryption from DDH. In: IACR Cryptol. ePrint Arch., p. 311 (2022)

- [NAL15] Nuñez, D., Agudo, I., López, J.: Ntrurencrypt: An efficient proxy re-encryption scheme based on NTRU. In: ACM ASIA CCS 2015 (2015)
- [Nis21] Nishimaki, R.: The direction of updatable encryption does matter. Cryptology ePrint Archive, Report 2021/221 (2021). <https://eprint.iacr.org/2021/221>
- [NS12] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. *SIAM J. Comput.* **41**(4), 772–814 (2012)
- [Nun18] Nunez, D.: Umbral: a threshold proxy re-encryption scheme. NuCypher Inc and NICS Lab, University of Malaga, Spain (2018)
- [NX15] Nishimaki, R., Xagawa, K.: Key-private proxy re-encryption from lattices, revisited. *IEICE Trans.* **98-A**(1), 100–116 (2015)
- [Pay18] Payment Card Industry (PCI). Data Security Standard - Version 3.2.1, May (2018)
- [PRSV17] Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, **20**(4):14:1–14:31 (2017)
- [PWA+16] Phong, L.T., Wang, L., Aono, Y., Nguyen, M.H., Boyen, X.: Proxy re-encryption schemes with key privacy from LWE. Cryptology ePrint Archive, Report 2016/327 (2016) <https://eprint.iacr.org/2016/327>
- [SD19] Sehrawat, V.S., Desmedt, Y.: Bi-homomorphic lattice-based PRFs and unidirectional updatable encryption. In: Mu, Y., Deng, R.H., Huang, X. (eds.) CANS 2019. LNCS, vol. 11829, pp. 3–23. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-31578-8\\_1](https://doi.org/10.1007/978-3-030-31578-8_1)
- [SS21] Slamanig, D., Striecks, C.: Puncture 'em all: Stronger updatable encryption with no-directional key updates. Cryptology ePrint Archive, Report 2021/268 (2021) <https://eprint.iacr.org/2021/268>



# Backward-Leak Uni-Directional Updatable Encryption from (Homomorphic) Public Key Encryption

Yao Jiang Galteland<sup>1</sup> and Jiaxin Pan<sup>2</sup>

<sup>1</sup> Department of Information Security and Communication Technology, NTNU – Norwegian University of Science and Technology, Gjøvik, Norway

yao.jiang@ntnu.no

<sup>2</sup> Department of Mathematical Sciences, NTNU – Norwegian University of Science and Technology, Trondheim, Norway

jiaxin.pan@ntnu.no

**Abstract.** The understanding of directionality for updatable encryption (UE) schemes is important, but not yet completed in the literature. We show that security in the backward-leak uni-directional key updates setting is equivalent to the no-directional one. Combining with the work of Jiang (ASIACRYPT 2020) and Nishimaki (PKC 2022), it is showed that the backward-leak notion is the strongest one among all known key update notions and more relevant in practice. We propose two novel generic constructions of UE schemes that are secure in the backward-leak uni-directional key update setting from public key encryption (PKE) schemes: the first one requires a key and message homomorphic PKE scheme and the second one requires a bootstrappable PKE scheme. These PKE can be constructed based on standard assumptions (such as the Decisional Diffie-Hellman and Learning With Errors assumptions).

**Keywords:** Updatable Encryption · Public Key Encryption · Backward-leak Uni-Directional Key Update · No-Directional Key Update · Standard Assumption

## 1 Introduction

To mitigate key compromise over time, a data subject wishes to periodically update her outsourced data on a data host. The outsourced data is expected to be refreshed from the old key to the new key without changing the underlying

---

Y. Jiang Galteland—Her work has been co-funded by the IKTPLUSS program of the Research Council of Norway under the scope of and as part of the outcome from the research project Reinforcing the Health Data Infrastructure in Mobility and Assurance through Data Democratization (Health Democratization, 2019–2024, Project No. 288856.

J. Pan—His work is supported by the Research Council of Norway under Project No. 324235.

© International Association for Cryptologic Research 2023

A. Boldyreva and V. Kolesnikov (Eds.): PKC 2023, LNCS 13941, pp. 399–428, 2023.

[https://doi.org/10.1007/978-3-031-31371-4\\_14](https://doi.org/10.1007/978-3-031-31371-4_14)

message. During this process, it is also reasonable to expect that no information of plaintexts are leaked while updating.

Updatable encryption (UE) schemes [2–4, 6, 11, 12, 14, 16] are a special kind of encryption schemes that allow the data host to update ciphertexts with the help of a data subject generated updating material, namely update token. Update tokens can rotate ciphertexts or keys, which makes UE schemes particularly interesting for outsourced data storage. However, leaked tokens together with key corruption an adversary may break confidentiality of the future or past epoch by upgrading or downgrading keys or ciphertexts, which is captured by the directionality of UE schemes. The study of UE mainly focuses on the security notions and efficient constructions. Directionality for UE schemes is important to study since it plays a central role in influencing the security result. The challenge is that there are two types of ciphertext update settings and four types of key update settings in the literature, and a combination of these settings results in eight different types of update settings for UE schemes to analyze.

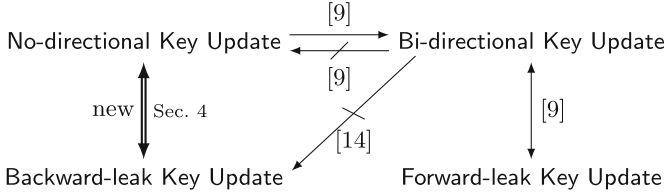
*Directionality of Ciphertext Updates.* If an update token can only update a ciphertext under a key in the past to a ciphertext under a new key without changing the encrypted message, it is in the forward direction, then we call that such a UE scheme has uni-directional ciphertext updates; and if an update token can additionally update a ciphertext to another ciphertext under a key in the past, we call such a UE scheme with bi-directional ciphertext updates.

*Directionality of Key Updates.* Secret key leakage is a serious security threat to encryption. For instance, there is no security guarantee for standard encryption schemes if their secret keys are leaked. However, for UE schemes, the update token offers a potential to preserve confidentiality, since the update token allows us to update a secret key and the corresponding ciphertexts. Realizing this fully is very challenging and requires careful treatments, since the update token may also leak information about the key.

Directionality of key update is used to capture which information adversaries can learn about the secret keys given the update token. Roughly speaking, there are four key update settings given by the literature [9, 12, 14]. For a precise description, let  $e$  be an epoch, namely, the index of a time period. In the *forward-leak* uni-directional key update setting<sup>1</sup> [12], given a key  $k_e$  and an update token  $\Delta_{e+1}$  adversaries can only learn a key  $k_{e+1}$  in the forward direction. Similarly, in the *backward-leak* uni-directional key update setting [14], adversaries can only learn a key  $k_e$  in the backward direction, given  $k_{e+1}$  and  $\Delta_{e+1}$ . If both forward-leak and backward-leak are satisfied in a setting, then it is called *bi-directional key update*. In contrary, if an update token leaks nothing about any secret key, then this setting is called *no-directional key update* [9].

*Security Implications Among Different Key Update Settings.* Security of a UE scheme is defined with respect to the aforementioned key update settings.

<sup>1</sup> This was called uni-directional key updates in [12], but here we follow the more precise terminology of Nishimaki [14] and call it forward-leak uni-directional key updates.



**Fig. 1.** Security implications among different key update settings assuming uni-directional ciphertext updates.  $X \rightarrow Y$  means that security in the  $X$  setting implies that in the  $Y$  setting, and  $X \not\rightarrow Y$  means that security in the  $X$  setting does not imply that in the  $Y$  setting, and  $X \leftrightarrow Y = (X \rightarrow Y) \wedge (Y \rightarrow X)$ . Contribution in this paper is marked with a double arrow ‘ $\leftrightarrow$ ’.

Roughly speaking, UE security guarantees confidentiality if the trivial win conditions are not triggered. The trivial win conditions are defined differently in each key update setting, and more information leaked about keys leads to more trivial win conditions in the confidentiality game for UE schemes. With more trivial win conditions, it seems harder for an adversary to win the confidentiality, since it is easier for it to trigger the trivial win conditions. Thus, intuitively, a setting with less key leakage seems to give stronger security.

This intuition partially holds true, according to the work of Jiang [9] and Nishimaki [14]. More precisely, in [9] it has been showed that security in the no-directional setting is strictly stronger than that in the bi-directional setting, and security in the bi-directional key update setting is equivalent to that in the forward-leak setting. To further complete the work of Jiang, Nishimaki [14] proposed the backward-leak uni-directional setting and showed that UE schemes in prior works [4,9,12] are secure in the bi-directional setting but insecure in the backward-leak uni-directional setting. Here we consider that an update token can only update a ciphertext in the forward direction, since if the ciphertext updates are bi-directional then all four settings are equivalent as shown in [9]. The implications among these four key update settings are shown figuratively in Fig. 1.

To sum up the discussions above, it is currently unclear that the relation between the no-directional and backward-leak uni-directional key update settings, although they both are stronger than the bi-directional and forward-leak setting.

*Our Goal: UE schemes with Strong Security from Weak Assumptions.* We aim at constructing UE schemes with strong security from weak assumptions. In achieving our goal, we first need to decide which notion is the strongest among the above four settings. Jumping ahead, our first contribution is proving the no-directional and backward-leak settings are equivalent. Given our equivalence result, we claim it is more desirable to construct a UE scheme that is secure in the backward-leak uni-directional key update setting for the following reasons.

Firstly, UE schemes secure in the backward-leak setting are technically more promising to construct based on weak assumptions, since the existing UE scheme [14] with no-directional key updates are based on strong assumptions. Namely,

the scheme in [14] requires a rather strong and impractical primitive, indistinguishability obfuscation.

Secondly, although there is a backward-leak UE scheme based on the Learning With Errors (LWE) assumption proposed by Nishimaki [14], it is unknown whether backward-leak UE schemes can be constructed from a wider class of weak assumptions, for instance, the Diffie-Hellman assumption without pairings. We are particularly interested in constructing UE schemes generically from public-key encryption (PKE), since this not only is theoretically interesting, but also can give us UE schemes from rather weak assumptions. Recently, Alapati, Montgomery, and Patranabis [1] have proved that ciphertext-independent UE implies PKE, but the implication in the other direction is unknown.

Finally, we stress that the backward-leak setting is relevant for practice, as discussed in [14]. In practice, the purpose of updating our keys is mostly because the current key and those in the past may be leaked. In such a scenario, UE schemes in the backward-leak uni-directional key update setting are required, since they can provide confidentiality in an epoch, even though all previous keys and tokens are corrupted. Moreover, backward-leak UE schemes remain secure even if the data host forgets to delete older keys and tokens, while this is not the case for forward-leak UE schemes, since with the older keys an adversary can learn the keys in the future.

## 1.1 Our Contributions

*Intuition Behind Security Definitions.* Our first contribution is providing an intuitive understanding of trivial win conditions, firewalls, directionality and security notions. Explanations of all these topics exists in [3, 4, 9, 11, 12, 14], however, we aim to provide a simple description to show the relations among these definitions. We consider two classes of UE schemes (discussed in Sect. 3.2 and 3.3): the first class of UE schemes have update settings such that keys cannot upgrade and ciphertexts cannot downgrade, the second class of UE schemes have update settings where keys and ciphertexts both can leak information in the forward direction. We observe that the first class of UE schemes (including UE schemes with backward-leak uni-directional key updates and no-directional key updates) can achieve the strongest confidentiality notion (with post-compromise security). Thus, it is only necessary to analyze two classes of UE schemes, and these two classes of UE schemes matches with the equivalence result in the literature [9, 14] and our work. That is, the eight variants of confidentiality notions can be seen as only two classes of confidentiality notions. We will show that notions in the same class are equivalent and one class is strictly stronger than the other.

*Equivalence Result.* Our second contribution is proving that security in the backward-leak uni-directional key update and uni-directional ciphertext update setting is equivalent to that in the no-directional key update and uni-directional ciphertext update setting. All our UE schemes have uni-directional ciphertext updates, and for simplicity, we do not mention it explicitly in the remaining of this section. This means that UE schemes with no-directional key updates do not



provide stronger security than UE schemes with backward-leak uni-directional key update. Our result suggests that constructing UE schemes with backward-leak uni-directional key update is equivalent to constructing UE schemes with no-directional key update.

*Generic Constructions of UE from PKE.* Our third contribution is constructing two generic constructions of UE schemes with backward-leak uni-directional key update from PKE schemes. Our constructions require additional properties of the underlying PKE schemes. Our first construction requires key and message homomorphism for PKE schemes. Such PKE schemes can be instantiated under the Decisional Diffie-Hellman (using the ElGamal encryption) and LWE (using the Regev encryption [15]) assumptions. Combining with our equivalence result, the aforementioned two schemes provide us with the *first* no-directional secure UE schemes without pairings in the Diffie-Hellman setting and based on a post-quantum assumption, respectively. We note that the uni-directional schemes from FHE or IO or lattice trapdoors [14] usually do not have this increased key-size or ciphertext-size. But without these assumptions and technique, uni-directional schemes relying on standard assumptions (namely, ours and the work in [13]) are constructed with growing key and ciphertext size. It remains an open problem to construct uni-directional schemes relying on standard assumptions where the key and ciphertext size keeps the same.

Our second generic construction uses a bootstrappable PKE [8] that can be implemented using the LWE assumption, which again gives us a post-quantum UE scheme with security in the no-directional key update setting.

Of independent interest, we propose a generic construction of bi-directional UE scheme from a key homomorphic PKE scheme. Our generic construction abstracts the constructions of RISE [12] and LWEUE [9]. We stress that our notion of key homomorphic PKE is inspired by the key homomorphic PRF of Boneh et al. [3] but different to theirs. More precisely, our key homomorphic property is defined with respect to a public key and a secret key, while theirs is with respect to two secret keys.

*Technical Overview.* Here we provide a brief technical overview of our generic backward-leak UE constructions from PKEs. The full descriptions of our schemes can be found in Sects. 5.3 and 6. The update token plays an important role in a UE scheme, and therefore we mostly focus on it in the following.

In our first construction, its key contains a pair of secret and public keys from the PKE scheme. The update token contains the difference between the old and new secret keys. In addition, the token includes an independently generated public key, which will play a central role for the confidentiality in the next epoch. To update a ciphertext, we have two steps: Firstly, by the key homomorphic property of the PKE, the difference between the old and new secret keys can be used to modify the ciphertext under the old key to one under the new key. Secondly, we randomize this ciphertext so that it is indistinguishable to a freshly generated ciphertext under the new key. In doing this, we use the aforementioned independent public key to encrypt a randomness to homomorphically randomize the ciphertext, since our PKE is further message homomorphic. In the security

proof, we can show that message is hidden by the randomness and confidentiality in the new epoch is preserved.

In our second construction, the update token is an encryption of the old secret key under the new public key. This token will not reveal any information about the new secret key even with the knowledge of the old secret key. To update a ciphertext, we evaluate the decryption circuit on the encryption of the old key (that is the token) and the encryption of the old ciphertext. The bootstrapping property states that this output is statistically close to a fresh ciphertext under the new key.

*Concurrent Work.* We note a recent work from Miao, Patranabis, and Watson [13] which has a construction of backward-leak uni-directional UE similar to ours, while our work contains a formal proof about the equivalence between backward-leak and no-directional UE.

## 1.2 More Discussion

*Ciphertext-Dependent v.s. Ciphertext-Independent.* We call an updatable encryption scheme is ciphertext-dependent [2,3,5,6] if the token generation process depends on the old ciphertext. If the token generation process is independent of the ciphertext to be updated, then the UE scheme is called ciphertext-independent [4,9,11,12,14,16]. A ciphertext-independent UE scheme is usually more efficient in terms of bandwidth cost, and, thus, we focus on such schemes in this paper.

*Deterministic Update.* The update algorithm in UE schemes can be deterministic or randomized. UE schemes with randomized update algorithm provide stronger security, in which the updated ciphertext can be in the same distribution of a fresh encryption. However, the work of Klooß et al. [11] shows that such UE schemes with randomized update cannot achieve ciphertext integrity and security against chosen-ciphertext attacks (CCA), but replayable CCA security. For instance, SHINE [4] and E&M [11] are UE schemes with deterministic update that have ciphertext integrity and CCA security. Klooß et al.'s result also means that our constructions cannot be CCA secure, but it is promising to make it RCCA secure. We leave constructing this as an open problem.

*Security Notions.* Boneh et al. [3] presented the first security notion for UE schemes. After that, the works in [4,9,11,12] proposed more realistic security notion by providing more capability to an adversary. For instance, it can adaptively corrupt epoch keys or update tokens at any point within the security game. Jiang [9] first discussed the directionality of security notions and showed that security notions with forward-leak uni- and bi-directional updates are equivalent in the current state-of-the-art UE security notion of Boyd et al. [4]. In addition, Jiang [9] proved that confidentiality notions with no-directional key updates are strictly stronger than uni- and bi-directional update variants of the corresponding notions. Nishimaki [14] defined a new type of key update, backward-leak uni-directional key update, which is not covered in the work of [9]. We will prove the

relation between the backward-leak uni-directional key update variant of a confidentiality notion with other update variants of the same confidentiality notion. We will show that confidentiality notions with backward-leak uni-directional key update is equivalent to no-directional key updates variants of the corresponding notions. Which means the backward-leak uni-directional key update variant of notions are the strongest notions.

Slamanig and Striecks [16] introduced a stronger model for UE schemes, where they consider an “expiry epoch”,  $e_{\text{exp}}$ . If a ciphertext is updated to an epoch  $e$ , where  $e \geq e_{\text{exp}}$  and  $e_{\text{exp}}$  is this ciphertext’s expiry epoch, then this ciphertext can no longer be decryptable. Their model allows the adversary knows “all”<sup>2</sup> tokens. Forward security is guaranteed if the key updates are at most in the forward-direction and leaked keys are in epochs after the expiry epoch of a ciphertext. Post compromise security is guaranteed if the key updates are at most in the backward-direction. Realizing such strong security strictly requires no-directional UE schemes where keys must not be updatable in any direction. Note that Jiang’s equivalence theorem [9] holds in the setting where there is no expiry date for ciphertexts, i.e.,  $e_{\text{exp}} = \infty$ . We consider the case for  $e_{\text{exp}} = \infty$  in this paper. We do not compare the efficiency of our construction with the UE construction in [16], due to schemes are in different UE models.

## 2 Preliminaries

In this section we describe the notation used in this paper and present the necessary background material of updatable encryption. Due to space limitations, we provide the preliminaries for public key encryption in the full version [7].  $\gamma$  parameter and  $\text{negl}$  denotes a negligible function. For distributions  $X$  and  $Y$ ,  $X \stackrel{s}{\approx} Y$  means  $X$  is statistically indistinguishable from  $Y$ .

### 2.1 Updatable Encryption and Confidentiality Notions

Updatable encryption (UE) schemes [3, 4, 11, 12] are a special kind of encryption schemes with an additional functionality where ciphertexts under one key can be transferred to ciphertexts under another key by an update token.

**Definition 1 (UE).** *An updatable encryption scheme UE is parameterized by a tuple of algorithms (Setup, Next, Enc, Dec, Upd) that operates over epochs such that*

- The setup algorithm  $\text{Setup}(\lambda)$  takes a security parameter  $\lambda$  as input, and outputs an initial epoch key  $k_1$ .
- The next algorithm  $\text{Next}(k_e)$  takes an epoch key  $k_e$  as input, and outputs a new key  $k_{e+1}$  and an update token  $\Delta_{e+1}$ , the update token can be used to update ciphertexts from epoch  $e$  to  $e + 1$ .

---

<sup>2</sup> except for some end epoch  $e_{\text{end}}$ , if  $e_{\text{exp}} \leq e_{\text{end}}$ .

- The encryption algorithm  $\text{Enc}(k_e, m)$  takes an epoch key  $k_e$  and a message  $m$  as input, and outputs a ciphertext  $c_e$ .
- The decryption algorithm  $\text{Dec}(k_e, c_e)$  takes an epoch key  $k_e$  and a ciphertext  $c_e$  as input, and outputs a message  $m$ .
- The update algorithm  $\text{Upd}(\Delta_{e+1}, c_e)$  takes an update token  $\Delta_{e+1}$  and a ciphertext  $c_e$  as input, and outputs an updated ciphertext  $c_{e+1}$ .

UE is correct if for any message  $m$ , any  $k_1 \leftarrow \text{Setup}(\lambda)$ , any  $(k_j, \Delta_j) \leftarrow \text{Next}(k_{j-1})$  for  $j = 2, \dots, e$ , and any  $c_i \leftarrow \text{Enc}(k_i, m)$  with  $i \in \{1, \dots, e\}$ , we have  $m = \text{Dec}(k_e, c_e)$  where  $c_j \leftarrow \text{Upd}(\Delta_j, c_{j-1})$  for  $j = i + 1, \dots, e$ .

Security notions for UE schemes include confidentiality and integrity. We do not consider integrity notions in this paper, see the paper by Jiang [9] for details. We review the confidentiality notion for UE schemes in Definition 2 and extend the six variants of security notions for UE schemes given by [9], in which the backward-leak uni-directional key update [14] variants are not included.

The confidentiality game is played between a challenger and an adversary, the adversary aims to distinguish a fresh encryption from an updated ciphertext. It is allowed for the adversary to adaptively corrupt keys and tokens, if any trivial win condition is triggered during the game the adversary will always lose. We will provide technical and high level understanding of trivial win conditions in Sect. 2.2 and 3.

**Definition 2** ([4,9]). Let  $\text{UE} = (\text{Setup}, \text{Next}, \text{Enc}, \text{Dec}, \text{Upd})$  be an updatable encryption scheme. Then the  $(\text{kk}, \text{cc})\text{-xxIND-UE-atk}$  advantage, for  $\text{kk} \in \{\text{no}, \text{f-uni}, \text{b-uni}, \text{bi}\}$ ,  $\text{cc} \in \{\text{uni}, \text{bi}\}$ ,  $\text{xx} \in \{\text{det}, \text{rand}\}$  and  $\text{atk} \in \{\text{CPA}, \text{CCA}\}$ , of an adversary  $\mathcal{A}$  against UE is defined as

$$\begin{aligned} \text{Adv}_{\text{UE}, \mathcal{A}}^{(\text{kk}, \text{cc})\text{-xxIND-UE-atk}}(\lambda) &= \left| \Pr[\text{Exp}_{\text{UE}, \mathcal{A}}^{(\text{kk}, \text{cc})\text{-xxIND-UE-atk-1}} = 1] - \Pr[\text{Exp}_{\text{UE}, \mathcal{A}}^{(\text{kk}, \text{cc})\text{-xxIND-UE-atk-0}} = 1] \right|, \end{aligned}$$

where the experiment  $\text{Exp}_{\text{UE}, \mathcal{A}}^{(\text{kk}, \text{cc})\text{-xxIND-UE-atk-b}}$  is given in Fig. 2.

## 2.2 Leakage Sets and Trivial Win Conditions

In this section, we review the definition of leakage sets and trivial win discussions [4,9,11,12], the leaked information can be used to help an adversary trivially win the confidentiality game.

**Trivial Win via Key and Ciphertext Leakage.** If an adversary knows both the key and the challenge-equal ciphertext in the same epoch period  $e$ , then the adversary can use this key to decrypt the challenge-equal ciphertext and obtain the underlying plaintext to win the confidentiality game. We use leakage sets to identify if this trivial win condition (“ $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ ”) is triggered.

Leakage sets [9,11,12] are defined to track epochs in which the adversary knows a key, a token, or learned a version of challenge ciphertext. The direct

$\text{Exp}_{\text{UE}, \mathcal{A}}^{\text{xxIND-UE-atk-b}} :$ $\text{do Setup; phase} \leftarrow 0$ $b' \leftarrow \mathcal{A}^{\text{oracles}}(\lambda)$ $\text{if } \left( (\mathcal{K}_{\text{kk}}^* \cap \mathcal{C}_{\text{kk,cc}}^* \neq \emptyset) \text{ or } (\text{xx} = \text{det and } (\tilde{e} \in \mathcal{T}_{\text{kk}}^* \text{ or } \mathcal{O}.\text{Upd}(\tilde{c}) \text{ is queried})) \right) \text{ then}$ $\quad \text{twf} \leftarrow 1$ $\text{if twf} = 1 \text{ then}$ $\quad b' \stackrel{\$}{\leftarrow} \{0, 1\}$ $\text{return } b'$ $\text{Setup}(\lambda)$ $k_1 \stackrel{\$}{\leftarrow} \text{Setup}(\lambda)$ $\Delta_1 \leftarrow \perp; e, c, \text{twf} \leftarrow 0$ $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$ $\mathcal{O}.\text{Enc}(m) :$ $c \leftarrow c + 1$ $c_e \stackrel{\$}{\leftarrow} \text{Enc}(k_e, m)$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(c, c_e, e; m)\}$ $\text{return } c_e$ $\mathcal{O}.\text{Dec}(c) :$ $m' \text{ or } \perp \leftarrow \text{Dec}(k_e, c)$ $\text{if } \left( (\text{xx} = \text{det and } (c, e) \in \tilde{\mathcal{L}}_{\text{kk,cc}}^*) \text{ or } (\text{xx} = \text{rand and } (m', e) \in \tilde{\mathcal{Q}}_{\text{kk,cc}}^*) \right) \text{ then}$ $\quad \text{twf} \leftarrow 1$ $\text{return } m' \text{ or } \perp$ $\mathcal{O}.\text{Next}() :$ $(\Delta_{e+1}, k_{e+1}) \leftarrow \text{Next}(k_e)$ $\text{if phase} = 1 \text{ then}$ $\quad \tilde{c}_{e+1} \leftarrow \text{Upd}(\Delta_{e+1}, \tilde{c}_e)$ $e \leftarrow e + 1$	$\mathcal{O}.\text{Upd}(c_{e-1}) :$ $\text{if } (j, c_{e-1}, e - 1; m) \notin \mathcal{L} \text{ then}$ $\quad \text{return } \perp$ $c_e \leftarrow \text{Upd}(\Delta_e, c_{e-1})$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, c_e, e; m)\}$ $\text{return } c_e$ $\mathcal{O}.\text{Corr}(\text{inp}, \hat{e}) :$ $\text{if } \hat{e} > e \text{ then}$ $\quad \text{return } \perp$ $\text{if inp} = \text{key then}$ $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{e}\}$ $\quad \text{return } k_{\hat{e}}$ $\text{if inp} = \text{token then}$ $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{e}\}$ $\quad \text{return } \Delta_{\hat{e}}$ $\mathcal{O}.\text{Chall}(\tilde{m}, \tilde{c}) :$ $\text{if phase} = 1 \text{ then}$ $\quad \text{return } \perp$ $\text{phase} \leftarrow 1; \tilde{e} \leftarrow e$ $\text{if } (\cdot, \tilde{c}, \tilde{e} - 1; \cdot) \notin \mathcal{L} \text{ then}$ $\quad \text{return } \perp$ $\text{if } b = 0 \text{ then}$ $\quad \tilde{c}_{\tilde{e}} \leftarrow \text{Enc}(k_{\tilde{e}}, \tilde{m})$ $\text{else}$ $\quad \tilde{c}_{\tilde{e}} \leftarrow \text{Upd}(\Delta_{\tilde{e}}, \tilde{c})$ $\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{e}\}$ $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{c}_{\tilde{e}}, \tilde{e})\}$ $\text{return } \tilde{c}_{\tilde{e}}$ $\mathcal{O}.\text{Upd}\tilde{\mathcal{C}} :$ $\text{if phase} \neq 1 \text{ then}$ $\quad \text{return } \perp$ $\mathcal{C} \leftarrow \mathcal{C} \cup \{e\}$ $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(c_e, e)\}$ $\text{return } \tilde{c}_e$
---	---

**Fig. 2.** The confidentiality experiment  $\text{Exp}_{\text{UE}, \mathcal{A}}^{(\text{kk,cc})\text{-xxIND-UE-atk-b}}$  for updatable encryption scheme UE and adversary  $\mathcal{A}$ , for  $\text{kk} \in \{\text{no, f-uni, b-uni, bi}\}$ ,  $\text{cc} \in \{\text{uni, bi}\}$ ,  $\text{xx} \in \{\text{det, rand}\}$  and  $\text{atk} \in \{\text{CPA, CCA}\}$ . The flag phase tracks whether or not  $\mathcal{A}$  has queried the  $\mathcal{O}.\text{Chall}$  oracle,  $\tilde{e}$  denotes the epoch in which the  $\mathcal{O}.\text{Chall}$  oracle happens, and twf tracks if the trivial win conditions are triggered. Oracles an adversary can query are  $\mathcal{O}.\text{Enc}$ ,  $\mathcal{O}.\text{Next}$ ,  $\mathcal{O}.\text{Upd}$ ,  $\mathcal{O}.\text{Corr}$ ,  $\mathcal{O}.\text{Chall}$  and  $\mathcal{O}.\text{Upd}\tilde{\mathcal{C}}$  if  $\text{atk} = \text{CPA}$ . If  $\text{atk} = \text{CCA}$ ,  $\mathcal{O}.\text{Dec}$  is included in the oracles. Leakage sets  $\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{K}_{\text{kk}}^*, \mathcal{T}_{\text{kk}}^*, \mathcal{C}_{\text{kk,cc}}^*, \tilde{\mathcal{L}}_{\text{kk,cc}}^*, \tilde{\mathcal{Q}}_{\text{kk,cc}}^*$  are discussed in Sect. 2.2.

leakage sets  $\mathcal{K}, \mathcal{T}, \mathcal{C}$  are describe as follows. Furthermore,  $\mathcal{K}^*, \mathcal{T}^*$  and  $\mathcal{C}^*$  are defined as the extended sets of  $\mathcal{K}, \mathcal{T}$  and  $\mathcal{C}$  to track the indirect leakage.

- $\mathcal{K}$ : Set of epochs in which the adversary corrupted the key (from  $\mathcal{O}.\text{Corr}$ ).
- $\mathcal{T}$ : Set of epochs in which the adversary corrupted the token (from  $\mathcal{O}.\text{Corr}$ ).
- $\mathcal{C}$ : Set of epochs in which the adversary learned a challenge-equal ciphertext<sup>3</sup> (from  $\mathcal{O}.\text{Chall}$  or  $\mathcal{O}.\text{Upd}\tilde{\mathcal{C}}$ ).

*Key Leakage.* The size of the key leakage set  $\mathcal{K}^*$  can be influenced by the key update direction of UE schemes. In the no-directional key update setting [9], the adversary does not have more information about keys except for set  $\mathcal{K}$ . In the forward-leak uni-directional key update setting, if the adversary knows a key  $k_e$  and an update token  $\Delta_{e+1}$  then it can infer the next key  $k_{e+1}$ . In the backward-leak [14] uni-directional key update setting, if the adversary knows a key  $k_{e+1}$  and an update token  $\Delta_{e+1}$  then it can infer the previous key  $k_e$ .

The notations **f-uni** and **b-uni** denote forward-leak uni and backward-leak uni, resp.. In the **kk**-directional key update setting, for  $\text{kk} \in \{\text{no}, \text{f-uni}, \text{b-uni}, \text{bi}\}$ , denote the set  $\mathcal{K}_{\text{kk}}^*$  as the extended set of corrupted key epochs. We compute these sets as follows, where the boxed part is only computed for  $\text{kk} \in \{\text{b-uni}, \text{bi}\}$ , the gray boxed part is only computed for  $\text{kk} \in \{\text{f-uni}, \text{bi}\}$

$$\begin{aligned} \mathcal{K}_{\text{kk}}^* &\leftarrow \{e \in \{0, \dots, l\} \mid \text{CorrK}(e) = \text{true}\} \\ \text{true} &\leftarrow \text{CorrK}(e) \iff \\ &(e \in \mathcal{K}) \vee \boxed{\text{CorrK}(e+1) \wedge e+1 \in \mathcal{T}} \vee \text{CorrK}(e-1) \wedge e \in \mathcal{T}. \end{aligned} \tag{1}$$

*Token Leakage.* The adversary directly learns all corrupted tokens, it can also compute a token from two consecutive epoch keys. We follow the assumption (an update token can be computed via two consecutive epoch keys) in the page 7 of [10], this assumption is essential to formulate the known knowledge to the adversary. Hence, for  $\text{kk} \in \{\text{no}, \text{f-uni}, \text{b-uni}, \text{bi}\}$ , denote  $\mathcal{T}_{\text{kk}}^*$  as the extended set of corrupted token epochs.

$$\mathcal{T}_{\text{kk}}^* \leftarrow \{e \in \{0, \dots, l\} \mid (e \in \mathcal{T}) \vee (e \in \mathcal{K}_{\text{kk}}^* \wedge e-1 \in \mathcal{K}_{\text{kk}}^*)\}. \tag{2}$$

*Challenge-Equal Ciphertext Leakage.* The adversary learned all versions of challenge ciphertexts in epochs in  $\mathcal{C}$ . Additionally, the adversary can compute challenge-equal ciphertexts via tokens. In the uni-directional ciphertext update setting, the adversary can upgrade ciphertexts. In the bi-directional ciphertext update setting, the adversary can additionally downgrade ciphertexts.

For  $\text{kk} \in \{\text{no}, \text{f-uni}, \text{b-uni}, \text{bi}\}$  and  $\text{cc} \in \{\text{uni}, \text{bi}\}$ , denote the set  $\mathcal{C}_{\text{kk},\text{cc}}^*$  as the extended set of challenge-equal epochs. We compute these sets as follows, where the boxed part is only computed for  $\text{cc} = \text{bi}$ .

$$\begin{aligned} \mathcal{C}_{\text{kk},\text{cc}}^* &\leftarrow \{e \in \{0, \dots, l\} \mid \text{ChallEq}(e) = \text{true}\} \\ \text{true} &\leftarrow \text{ChallEq}(e) \iff \\ &(e \in \mathcal{C}) \vee (\text{ChallEq}(e-1) \wedge e \in \mathcal{T}_{\text{kk}}^*) \vee \boxed{\text{ChallEq}(e+1) \wedge e+1 \in \mathcal{T}_{\text{kk}}^*}. \end{aligned} \tag{3}$$

---

<sup>3</sup> A challenge-equal ciphertext is either a challenge ciphertext or an updated ciphertext of the challenge ciphertext.

**Trivial Win Due to Deterministic Update.** In a confidentiality game with deterministic update setting. If the adversary knows the updated version (by either knowing the update token  $\Delta_{\mathbf{e}}$  or asking for an update oracle  $\mathcal{O}.\text{Upd}$  on  $\bar{c}$ ) of the challenge input ciphertext  $\bar{c}$ , it can compare the updated ciphertext with the challenge ciphertext to win the confidentiality game. This trivial win condition is “ $\bar{\mathbf{e}} \in \mathcal{T}^*$  or  $\mathcal{O}.\text{Upd}(\bar{c})$  is queried”.

**Trivial Wins via Decryption.** If the adversary submits a challenge-equal ciphertext to the decryption oracle, it can trivially win the confidentiality game by comparing the challenge plaintexts with the decryption output. Hence, the adversary is not allowed to ask for a decryption oracle on such ciphertexts. We use the following sets to track challenge ciphertexts, challenge plaintexts and their updated versions that can be known to the adversary. These sets can be used to identify the above trivial win condition. More precisely, “ $(c, \mathbf{e}) \in \tilde{\mathcal{L}}^*$ ” is a trivial win condition that is checked by the decryption oracle in the  $\text{detIND-UE-CCA}$  game, “ $(m', \mathbf{e}) \in \tilde{\mathcal{Q}}^*$ ” is a trivial win condition that is checked by the decryption oracle in the  $\text{randIND-UE-CCA}$  game.

- $\tilde{\mathcal{L}}^*$ : Set of challenge-equal ciphertexts  $(\bar{c}_{\mathbf{e}}, \mathbf{e})$ . The adversary learned these ciphertexts from  $\mathcal{O}.\text{Chall}$  or  $\mathcal{O}.\text{Upd}\bar{c}$ , or derived these ciphertexts from tokens.
- $\tilde{\mathcal{Q}}^*$ : Set of challenge plaintexts  $\{(\bar{m}, \mathbf{e}), (\bar{m}_1, \mathbf{e})\}$ , where  $(\bar{m}, \bar{c})$  is the input of challenge query  $\mathcal{O}.\text{Chall}$  and  $\bar{m}_1$  is the underlying message of  $\bar{c}$ . The adversary learned or was able to compute a challenge-equal ciphertext in epoch  $\mathbf{e}$  with the underlying message  $\bar{m}$  or  $\bar{m}_1$ .

### 3 Intuitions Behind Security Definitions

In this section, we propose some high level intuitions behind the security notions for UE.

We aim to clarify the relations among trivial win conditions, directionality and security results. If a UE scheme potentially leaks more information (which can be influenced by directionality), then there exists more vulnerabilities (trivial win conditions) for such scheme. Forward security in prior work of UE are achieved under the limitation that no trivial win condition is triggered, namely, there may exist some token after the challenge epoch cannot be corrupted. In our work, we define a relaxed version of forward security that, after the challenge epoch, any keys and tokens can be corrupted and the confidentiality of the challenge epoch remains. Similarly, we discussed post-compromise security. In the end, we provide some observations about what types of UE schemes can achieve forward or post-compromise security.

### 3.1 Intuition of Trivial Wins Conditions

The more information that get leaked in a confidentiality game the more chances the adversary gains to win that game, and trivial win conditions are defined to exclude such winning conditions. Generally speaking, the more update directions a UE scheme has the more information the adversary can infer. That is, the directionality of the update setting influences how much information gets leaked.

Trivial win conditions can be seen as a way of limiting the adversary's attacking power and this can be used to compare two notions, where two notions for UE schemes are equivalent if they have the same winning probability. Consider a modified confidentiality game where the adversary is not allowed to perform certain actions that will trigger trivial win conditions. If such action happens, the game aborts. The winning probability of the original confidentiality game is the same as this modified confidentiality game. In other words, trivial win conditions are equivalent to an attacking model. Such attacking model defines the restriction to the adversary, for example, the adaptive corruption ability is restricted by not triggering the trivial win conditions. Only when the adversary's attack actions does not trigger the trivial win conditions it is possible to win the game. Informally, a security notion for a system can be seen as stronger if the system remains secure even if the adversary has more attacking ability.

Combining this with the discussion above, we see that the more update direction a UE scheme has the more key and ciphertext leakage there will be and more trivial win conditions to evaluate. This means that the adversary will be limited more in such a confidentiality game. From a security point of view, the less attacking ability the adversary has the weaker we can regard a security notion.

### 3.2 Intuition of Firewalls

The observation of firewalls was introduced in the work of Lehmann and Tackmann [12], Kloof et al. [11] provided an extended description of this *key insulation* technique, and Boyd et al. [4] formally defined it as *firewall technique*. Furthermore, Nishimaki [14] proposed a *relaxed firewall*, where the token on the left side of the insulated region ( $\Delta_{\text{fwl}}$ ) can be corrupted. In any security game, if the adversary never triggers the trivial win conditions, a cryptographic separation (firewalls) exists, for a detailed discussion of the existence of firewalls see [4, 11, 12, 14].

**Definition 3 (Firewalls [4, 11, 12]).** *An insulated region with firewalls  $\text{fwl}$  and  $\text{fwr}$  is a consecutive sequence of epochs  $(\text{fwl}, \dots, \text{fwr})$  for which:*

- no key in the sequence of epochs  $(\text{fwl}, \dots, \text{fwr})$  is corrupted;
- the tokens  $\Delta_{\text{fwl}}$  and  $\Delta_{\text{fwr}+1}$  are not corrupted (if they exist);
- all tokens  $(\Delta_{\text{fwl}+1}, \dots, \Delta_{\text{fwr}})$  are corrupted (if any exist).



**Definition 4 (Relaxed Firewalls [14]).** A relaxed insulated region *with* relaxed firewalls  $\text{fwl}$  and  $\text{fwr}$  is a consecutive sequence of epochs  $(\text{fwl}, \dots, \text{fwr})$  for which:

- no key in the sequence of epochs  $(\text{fwl}, \dots, \text{fwr})$  is corrupted;
- the token  $\Delta_{\text{fwr}+1}$  is not corrupted (if it exists);
- all tokens  $(\Delta_{\text{fwl}+1}, \dots, \Delta_{\text{fwr}})$  are corrupted (if any exist).

The firewall technique is used when proving the security for UE schemes, it provides a method of describing a cryptographic separation, which is required in the epoch based model to simulate where keys and ciphertexts are known or unknown to the adversary. Firewalls, and relaxed firewalls, define a “safe” or insulated region for keys, where no key within the region can be inferred from keys outside of this region. We can regard the tokens  $\Delta_{\text{fwl}}$  and  $\Delta_{\text{fwr}+1}$  as the left and right firewalls, the cryptographic separation is created when these two tokens are unknown to the adversary. In some UE settings, the token  $\Delta_{\text{fwl}}$  can be corrupted and the cryptographic separation still holds, which means that the insulated region can be relaxed even without the left firewall.

The relaxed insulated region is suitable for analyzing security for UE schemes with update settings such that keys cannot upgrade and ciphertexts cannot downgrade, namely ciphertext and key will not leak information in the same direction. In UE schemes with uni-directional ciphertext update setting and key update settings without forward-leak direction, we have that the token  $\Delta_{\text{fwl}}$  cannot upgrade early epoch keys to learn keys inside the insulated region, it cannot downgrade challenge-equal ciphertexts to learn early challenge-equal ciphertexts outside of the insulated region as well. Hence, keys and ciphertexts inside and outside of the insulated region are separated even when  $\Delta_{\text{fwl}}$  is known to the adversary, the insulated region can be relaxed to allow the token  $\Delta_{\text{fwl}}$  to be corrupted.

The (original) insulated region is suitable for analyze UE schemes with update settings such that keys and ciphertexts can both leak information in the forward direction, namely ciphertext and key will leak information in the same direction. In UE schemes with ciphertext and key update settings that both have at least forward-leak direction, we have that the token  $\Delta_{\text{fwl}}$  can upgrade early corrupted keys to learn keys inside the insulated region. For these UE schemes we have that the token  $\Delta_{\text{fwr}}$  can upgrade challenge-equal ciphertexts to an epoch outside the insulated region where the adversary knows a corrupted key. Hence, both tokens  $\Delta_{\text{fwl}}$  and  $\Delta_{\text{fwr}+1}$  are required to be unknown to the adversary to make a cryptographic separation.

### 3.3 Forward Security and Post-compromise Security

Forward and post-compromise security for UE schemes were discussed by Lehmann and Tackmann [12]. However, all confidentiality games in their work are restricted by not triggering trivial win conditions, which means there exists a cryptographic separation between the leaked key region and the “safe” region

(see the discussion of cryptographic separation in Sect. 3.2). That is, there exists two tokens one before and one after the epoch where the adversary aims to break the confidentiality such that these tokens cannot be corrupted.

We consider the standard definitions for forward and post-compromise security, in which we do not have the restrictions of tokens which cannot be corrupted. We say a UE scheme have

- *forward security* if the confidentiality in early epochs are not broken even if an adversary compromises keys and tokens in some later epochs.
- *post-compromise security* if the confidentiality in later epochs are not broken even if an adversary compromises keys and tokens in some early epochs.

We observe that only some specific UE schemes can achieve post-compromise security. No UE scheme can achieve forward security.

The first class of UE schemes, discussed in Sect. 3.2, have update settings such that keys cannot upgrade and ciphertexts cannot downgrade. Since the ciphertext update is forward direction, an adversary can upgrade challenge-equal ciphertexts by the help of tokens to an epoch where the adversary knows a key to break the confidentiality in early epochs. Therefore, such UE schemes cannot achieve forward security. However, an adversary compromises keys and tokens in some early epochs cannot learn keys to break the confidentiality in later epochs. Additionally, the adversary cannot downgrade a challenge-equal ciphertext to an early epoch where the adversary knows a key to break the confidentiality in later epochs. Hence, such UE schemes can have post-compromise security.

The second class of UE schemes, discussed in Sect. 3.2, have update settings where keys and ciphertexts both can leak information in the forward direction. An adversary can infer keys by tokens in the forward direction to break the confidentiality in later epochs, therefore, such UE schemes cannot achieve post-compromise security. Moreover, since the ciphertext update is forward direction, similar to the discussion in above paragraph, such UE schemes cannot achieve forward security either.

The above discussion matches with the equivalence results of confidentiality notions (see Sect. 4). It implies that the first class of UE schemes (including UE schemes with backward-leak uni-directional key updates and no-directional key updates) can achieve the strongest confidentiality notion (with post-compromise security).

### 3.4 Directionality for UE Schemes and Confidentiality Notions

We specify that directionality for UE schemes and directionality for confidentiality notions are two different concepts. The update directionality for UE schemes was defined to measure how much keys and ciphertexts are leaked because of update tokens. However, such leakage can be the whole information leakage or just partial information leakage. The partial leakage is not captured in the directionality for UE schemes, there is no definition for partial information leakage. Under current definitions, we consider update direction for UE schemes to be

without partial information leakage. However, partial information leakage may be used to break the confidentiality. An adversary may be able to decrypt ciphertexts where it only knows partial information about the corresponding key. Such partial information leakage is considered in the security notion and it can be seen as equivalent to the whole information leakage. For example, suppose tokens in a UE scheme can be used to infer partial key information in both update direction, such UE schemes are considered as a no-directional key update UE scheme. However, if an adversary can use this partially leaked key to break confidentiality, then such UE scheme is not secure in the no-directional update variant of confidentiality, it can at most be secure in the bi-directional update variant of confidentiality. Overall, UE schemes with one kind of update setting do not immediately have the same update direction variant of security.

A specific update variant of security notion is suitable for examining UE schemes with the same update setting. But we stress that our security definitions are not restricted to UE schemes with some particular update setting, such UE schemes are simply insecure in a less updating (or leakage) variant of notion.

## 4 Relations Among Confidentiality Notions

In the work of [10], Jiang showed that all variants of the same integrity notions are equivalent, hence, we do not consider integrity notions in this work and focus on discussing the relations among confidentiality notions in this section. We prove that the backward-leak uni- and no directional key update variant of the same confidentiality notion are equivalent. As a result, UE schemes with backward-leak uni-directional key updates is as strong as UE schemes with no directional key update. It implies that we can construct a less hard (backward-leak uni-directional key updates) UE scheme to achieve the same security result.

### 4.1 Equivalence for Trivial Win Conditions

We prove four equivalence of the trivial win conditions. The proofs of these equivalence lemmas follow the proof strategy in [9], and they are provided in the full version [7]. The trivial win conditions considered in this section are checked in a confidentiality game. In conclusion, if the trivial win conditions in the backward-leak uni-directional key update setting are triggered then the same trivial win conditions in the no directional key update setting would be triggered. We will use these trivial win equivalences to prove the relation in Theorem 1.

**Lemma 1 (Equivalence for Trivial Win Condition “ $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ ”).** *For any sets  $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, l\}$ , we have  $\mathcal{K}_{\text{b-uni}}^* \cap \mathcal{C}_{\text{b-uni,uni}}^* \neq \emptyset \iff \mathcal{K}_{\text{no}}^* \cap \mathcal{C}_{\text{no,uni}}^* \neq \emptyset$ .*

**Lemma 2 (Equivalence for Trivial Win Condition “ $\tilde{e} \in \mathcal{T}^*$  or  $\mathcal{O}.\text{Upd}(\bar{c})$  is queried”).** *For any  $\mathcal{K}, \mathcal{T}, \mathcal{C}$ . Suppose  $\mathcal{K}_{\text{kk}}^* \cap \mathcal{C}_{\text{kk,cc}}^* = \emptyset$ , where  $\text{kk} \in \{\text{b-uni, no}\}$ ,  $\text{cc} = \text{uni}$ , then*

$$\tilde{e} \in \mathcal{T}_{\text{no}}^* \text{ or } \mathcal{O}.\text{Upd}(\bar{c}) \text{ is queried} \iff \tilde{e} \in \mathcal{T}_{\text{b-uni}}^* \text{ or } \mathcal{O}.\text{Upd}(\bar{c}) \text{ is queried.}$$

**Lemma 3 (Equivalence for Trivial Win Condition “ $(c, e) \in \tilde{\mathcal{L}}^*$ ”).** For any sets  $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, e\}$ . Suppose  $\mathcal{K}_{b\text{-uni}}^* \cap \mathcal{C}_{b\text{-uni,uni}}^* = \emptyset$ , then

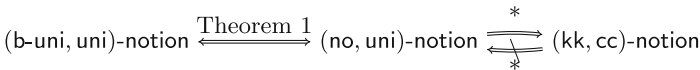
$$(c, e) \in \tilde{\mathcal{L}}_{b\text{-uni,uni}}^* \iff (c, e) \in \tilde{\mathcal{L}}_{no,uni}^*$$

**Lemma 4 (Equivalence for Trivial Win Condition “ $(m', e) \in \tilde{\mathcal{Q}}^*$ ”).** For any sets  $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, e\}$ .  $\mathcal{K}_{b\text{-uni}}^* \cap \mathcal{C}_{b\text{-uni,uni}}^* = \emptyset$ , then  $(m', e) \in \tilde{\mathcal{Q}}_{b\text{-uni,uni}}^* \iff (m', e) \in \tilde{\mathcal{Q}}_{no,uni}^*$ .

### 4.2 Relations Among Confidentiality Notions

Jiang [10] showed that the bi-directional key update and the forward-leak uni-directional key update variants of the same confidentiality notions are equivalent, and confidentiality in the no-directional key update is strictly stronger than that in the above mentioned two key update settings. However, the relation between the backward-leak uni-directional key update and the no-directional key update variants of the same confidentiality notion is missing in the previous work. We complete the relations among the eight variants of the same confidentiality notion, they are described as in Fig. 3. This is proven via Theorem 1, given below, and results in [10].

Recall discussions in Sect. 3.2 and 3.3, where we consider update settings in  $\{(b\text{-uni}, uni), (no, uni)\}$  are in one class and the rest update settings are in another class. Intuitive observation shows notions in the same class are equivalent and the prior class is strictly stronger than the latter. Interestingly, the result showed in Fig. 3 matches with this intuition and provides a rigorous proof.



**Fig. 3.** Relations among the eight variants of the same confidentiality notion, where notion  $\in \{\text{detIND-UE-CPA}, \text{randIND-UE-CPA}, \text{detIND-UE-CCA}, \text{randIND-UE-CCA}\}$ ,  $kk \in \{no, b\text{-uni}, f\text{-uni}, bi\}$  and  $cc \in \{uni, bi\}$ , the notation of  $(kk, cc)$  are except for values in  $\{(b\text{-uni}, uni), (no, uni)\}$ . Results in work of [10] are marked with \*.

In Theorem 1, we compare two types of UE notions: the no-directional key update setting and the backward-leak uni-directional setting. To illustrate the intuition for our equivalence between these two settings we have two scenarios: the first is where the adversary has corrupted a key located in an epoch earlier than the challenge epoch, and the second is where the adversary has corrupted a key located in an epoch later than the challenge epoch. In the first scenario, the adversary cannot update the key even if she had all update token because in both update settings forward key update is impossible. Similarly, the challenge ciphertext cannot be downgraded. Thus, in the first scenario, both update settings are equivalent. For the second scenario, we have that the adversary will win

the game in both update settings if she has access to enough tokens, because now the key can either be downgraded, using the tokens, or the challenge ciphertext can be upgraded, using the same tokens. So, in the second scenario, both update settings are equivalent.

**Theorem 1.** *Let  $UE = (\text{Setup}, \text{Next}, \text{Enc}, \text{Dec}, \text{Upd})$  be an updatable encryption scheme and  $\text{notion} \in \{\text{detIND-UE-CPA}, \text{randIND-UE-CPA}, \text{detIND-UE-CCA}, \text{randIND-UE-CCA}\}$ . For any  $(\text{b-uni}, \text{uni})$ -notion adversary  $\mathcal{A}$  against UE, there exists a  $(\text{no}, \text{uni})$ -notion adversary  $\mathcal{B}_1$  against UE such that*

$$\text{Adv}_{UE, \mathcal{A}}^{(\text{b-uni}, \text{uni})\text{-notion}}(\lambda) = \text{Adv}_{UE, \mathcal{B}_1}^{(\text{no}, \text{uni})\text{-notion}}(\lambda).$$

*Proof.* The proof follows the same method as the proof of Theorem 3.1 in [10]. We construct a reduction  $\mathcal{B}_1$  running the  $(\text{no}, \text{uni})$ -notion experiment which will simulate the responses of queries made by the  $(\text{b-uni}, \text{uni})$ -notion adversary  $\mathcal{A}$ . The reduction will send all queries received from  $\mathcal{A}$  to its  $(\text{no}, \text{uni})$ -notion challenger, and forwarding the responses to  $\mathcal{A}$ . Eventually, the reduction receives a guess from  $\mathcal{A}$  and forwards it to its own challenger. In the end, the  $(\text{no}, \text{uni})$ -notion challenger evaluates whether or not the reduction wins, if a trivial win condition was triggered the reduction is considered as losing the game. This final win evaluation will be passed to the adversary  $\mathcal{A}$ .

By the equivalences of trivial win conditions in Sect. 4.1 (Lemma 1 to 4), if  $\mathcal{A}$  does not trigger the trivial win conditions in the  $(\text{b-uni}, \text{uni})$ -notion game, then the reduction will not trigger the trivial win conditions in the  $(\text{no}, \text{uni})$ -notion game either. If  $\mathcal{A}$  triggers the trivial win conditions in the  $(\text{b-uni}, \text{uni})$ -notion game, then the reduction will also trigger the trivial win conditions in the  $(\text{no}, \text{uni})$ -notion game. Therefore, the reduction perfectly simulates the responses to adversary  $\mathcal{A}$ . And we have  $\text{Adv}_{UE, \mathcal{B}_1}^{(\text{no}, \text{uni})\text{-notion}}(\lambda) = \text{Adv}_{UE, \mathcal{A}}^{(\text{b-uni}, \text{uni})\text{-notion}}(\lambda)$ .

## 5 UE from Key Homomorphic PKE

We use key homomorphic PKE schemes to construct UE schemes in this section. The idea of key homomorphic by Boneh et al. [3] inspired this construction idea. Our key homomorphic is more general than the key homomorphic construction presented by Boneh et al. [3], where we can rotate information based on a public value and a secret value instead of two secret values.

### 5.1 Key Homomorphic PKE

We define key homomorphic PKE in this section, which will be used to build updatable encryption schemes in the later two sections.

**Definition 5 (Key Homomorphic PKE).** *We say public key encryption  $\text{PKE} = (\text{KG}, \text{Enc}, \text{Dec})$  is key homomorphic if:*

1. there exists an efficiently computable secret key to public key algorithm  $[\cdot] : SK \rightarrow PK$  such that for any  $(sk, pk) \leftarrow KG(\lambda)$ , the following two pairs of distributions are statistically close:

$$(sk, [sk]) \stackrel{s}{\approx} (sk, pk). \tag{4}$$

$$\{(sk_1 \otimes sk_2, [sk_1 \otimes sk_2]) \mid sk_1, sk_2 \stackrel{s}{\leftarrow} SK\} \stackrel{s}{\approx} \{(sk, [sk]) \mid sk \stackrel{s}{\leftarrow} SK\}. \tag{5}$$

where  $\otimes$  is an operation (which can be addition, multiplication, etc.) over the secret key space.

2. there exists an efficiently computable key homomorphic to key algorithm defined as:  
 $KHK : SK \times PK \rightarrow PK$  takes a secret key and a public key as input and outputs a public key, such that for any secret key  $sk_2 \in SK$  and public key  $pk_1 \in PK$ , the following two distributions are statistically close:

$$KHK(sk_2, pk_1) \stackrel{s}{\approx} [sk_1 \otimes sk_2], \tag{6}$$

where  $sk_1$  is the secret key of  $pk_1$ .

3. there exists an efficiently commutable key homomorphic to ciphertext algorithm  $KHC$ , defined as:  
 $KHC : SK \times CS \rightarrow CS$  takes a secret value and a ciphertext as input and outputs a ciphertext, such that for any keys  $(sk_1, pk_1) \leftarrow KG$ , secret value  $sk_2 \stackrel{s}{\leftarrow} SK$ , and any message  $m$ , the following two distributions are statistically close:

$$(c, KHC(sk_2, c)) \stackrel{s}{\approx} (c, Enc([sk_1 \otimes sk_2], m)), \tag{7}$$

where  $c = Enc(pk_1, m)$ .

*Remark 1.* Equation (4) guarantees that we can compute a public key from a secret key. Equation (5) makes sure the distribution generated from the homomorphism of keys is statistically close to the original key distribution. Equation (6) allows us to compute a new public key from a secret key (assume  $sk_2$ ) and an old public key (assume  $pk_1 = [sk_1]$ ), the underlying secret key of the newly generated public key matches the output of the homomorphic operation applied to the input secret keys ( $sk_1$  and  $sk_2$ ). It is essential for our security proof of Theorem 2. We need an algorithm to simulate new public keys from the corresponding old public keys and some secret values. In the proof of Theorem 2, when we use a reduction to simulate the game to an adversary within the firewall, the reduction has no knowledge of any secret keys, but it can simulate public keys with  $KHK$  from its own challenge public key and simulated secrets (cf. Fig. 11 in the full version [7]). Without these simulated public keys, the reduction cannot simulate valid ciphertexts within the firewall region. Additionally, we require the two distributions to be statistically close to make sure any adversary cannot distinguish simulated public keys from the real ones. All the PKE considered here

satisfy this property. This statistical property is crucial, since the correctness of our scheme requires Equation (7) to hold statistically. This, in turn, requires the two public keys are statistically indistinguishable, since otherwise Equation (7) cannot hold for an unbounded adversary. Equation (7) enables us to compute a new ciphertext from one public key to another public key without changing the underlying message.

Examples of key homomorphic public key encryption schemes are ElGamal encryption and LWE-based PKE (for some parameter choice) schemes. The security of such PKE schemes are based on DDH and LWE problems, resp..

- ElGamal encryption: Let  $\mathbb{G}$  be a cyclic group of order  $q$  with generator  $g$ . For any secret key  $x \in \mathbb{Z}_q$ , the corresponding public key is  $[x] = g^x$ . Given any public value  $[y]$  and any secret value  $x$ , we can compute  $\text{KHK}(x, [y]) = [y]^x = g^{xy} = [x \otimes y]$ . Here,  $\otimes$  is multiplication of integers and function  $\text{KHK}$  is computed as  $\text{KHK}(x, y) = y^x$ . For any ciphertext  $c = ([rx], [r] \cdot m)$ , any  $(y, [y]) \xleftarrow{\$} \text{KG}$  and a random value  $r' \xleftarrow{\$} \mathbb{Z}_q^*$ , note that  $[xy] = \text{KHK}(y, [x]) = [x]^y$ , then  $\text{KHC}(y, c) = (c_1^y \cdot [xy]^{r'}, [r'] \cdot c_2) = ([ (r + r')xy ], [r + r'] \cdot m)$  is a ciphertext encrypted under public key  $[xy]$  with the same underlying message  $m$  of the original ciphertext  $c$ .
- Lattice based PKE: Denote  $[s] = as + e$ , then we can compute  $\text{KHK}(s, [t]) = [s] + [t] \approx [s \otimes t]$ , the right side public key has a bigger error. Here,  $\otimes$  is group addition. For any ciphertext  $c = (ar, [s]r + e' + m)$  and any  $(t, [t]) \xleftarrow{\$} \text{KG}$ , note that  $[s \otimes t] \approx [s] + [t]$ , then  $\text{KHC}(t, c) = (c_1 + ar', c_2 + c_1t + [s \otimes t]r' + e'') \approx (a(r + r'), [s \otimes t](r + r') + (e' + e'') + m)$  is a ciphertext encrypted under public key  $[s \otimes t]$ .

### 5.2 Bi-directional UE from Key Homomorphic PKE

We construct a UE scheme PKEUE, which is constructed from a key homomorphic PKE scheme PKE, the construction is described in Fig. 4. Note that the next key pair  $(sk_{e+1}, pk_{e+1})$  is statistically close to a real key pair generated from  $\text{PKE.KG}(\lambda)$  by Eq. (4) and (5). Note that RISE [12] and LWEUE [9] are constructed by this method, they are built from ElGamal encryption and LWE-based PKE.

*Correctness.* The correctness of PKEUE follows from the correctness of PKE. It is sufficient to prove that the updated ciphertext is a valid ciphertext which keeps the same underlying message as the original ciphertext. Since PKE is key homomorphic (see Definition 5), we have that  $c_{e+1} = \text{PKE.KHC}(\Delta_{e+1}, c_e) \stackrel{\$}{\approx} \text{PKE.KHC}(\Delta_{e+1}, c_e)$  Equation (7)

$$\text{PKE.Enc}([sk_e \otimes \Delta_{e+1}], m) = \text{PKE.Enc}(pk_{e+1}, m).$$

Next, we prove that the UE scheme constructed above satisfies the weaker variant of rand-IND-UE security, namely the (bi, bi)-rand-IND-UE security.

$\frac{\text{Setup}(\lambda) :}{(sk_1, pk_1) \leftarrow \text{PKE.KG}(\lambda)}$ $\text{return } (sk_1, pk_1)$	$\frac{\text{Enc}(pk_e, m) :}{c_e \leftarrow \text{PKE.Enc}(pk_e, m)}$ $\text{return } c_e$
$\frac{\text{Next}(sk_e) :}{(\Delta_{e+1}, [\Delta_{e+1}]) \leftarrow \text{PKE.KG}(\lambda)}$ $sk_{e+1} \leftarrow sk_e \otimes \Delta_{e+1}$ $pk_{e+1} \leftarrow [sk_{e+1}]$ $\text{return } \Delta_{e+1}, (sk_{e+1}, pk_{e+1})$	$\frac{\text{Dec}(sk_e, c_e) :}{m' \leftarrow \text{PKE.Dec}(sk_e, c_e)}$ $\text{return } m'$
	$\frac{\text{Upd}(\Delta_{e+1}, c_e) :}{c_{e+1} \leftarrow \text{PKE.KHC}(\Delta_{e+1}, c_e)}$ $\text{return } c_{e+1}$

**Fig. 4.** PKEUE = (Setup, Next, Enc, Dec, Upd) is a UE scheme constructed from a key homomorphic PKE PKE.

**Theorem 2.** *Let PKEUE be the updatable encryption scheme described in Fig. 4. For any (bi, bi)-rand-IND-UE adversary  $\mathcal{A}$  against PKEUE, there exists an IND-CPA adversary  $\mathcal{B}_2$  against PKE such that*

$$\mathbf{Adv}_{\text{PKEUE}, \mathcal{A}}^{(\text{bi, bi})\text{-randIND-UE-CPA}}(\lambda) \leq l^3 \cdot \mathbf{Adv}_{\text{PKE}, \mathcal{B}_2}^{\text{IND-CPA}} + \text{negl}(\lambda),$$

where  $l$  is the upper bound on the last epoch.

*Proof.* In this proof, we use three steps to reach our desired goal. In the first step, we play a hybrid game over epochs, where the reduction constructs one hybrid for each epoch. In hybrid  $i$ , to the left of epoch  $i$  the game returns an updated ciphertext as the challenge output, to the right of epoch  $i$  it gives an encryption of the challenge input message as output. Therefore, we can move the (bi, bi)-randIND-UE-CPA game from left to right across the epoch space. In the second step, we apply the firewall technique [4, 11, 12] (recall the discussion in Sect. 3.2) so that we can construct a reduction (in step 3) playing the IND-CPA game by simulating the hybrid game to the adversary. Due to space limitations, the detailed security proof is shown in the full version [7].

### 5.3 Uni-Directional UE from Key and Message Homomorphic PKE

In this section, we construct a UE scheme with backward-leak uni-directional key update, which is called UNIUE. The UNIUE scheme is built from a key and message homomorphic PKE scheme. Recall that key homomorphic PKE is defined in Definition 5. The message homomorphic PKE is defined as the standard homomorphic encryption, we name it message homomorphic to distinguish two types of homomorphism (key homomorphism and message homomorphism) in this paper.

**Definition 6 (Message Homomorphic PKE).** *We say public key encryption PKE = (KG, Enc, Dec) is message homomorphic if for any message  $m_1, m_2 \in \mathcal{M}$  and any public key  $pk$ ,  $\text{Enc}(pk, m_1) \otimes \text{Enc}(pk, m_2) = \text{Enc}(pk, m_1 \oplus m_2)$ , where  $\otimes$  is an operation over the ciphertext space and  $\oplus$  is an operation over the message space.*



*Notation.* For a vector  $\mathbf{A} = (a_1, \dots, a_n)$ , we define  $[\mathbf{A}] = ([a_1], \dots, [a_n])$ . For vectors  $\mathbf{A}, \mathbf{B}$  and function  $f$ , we define  $f(\mathbf{A}, \mathbf{B}) = (f(a_1, b_1), \dots, f(a_n, b_n))$ .

*Constructing Uni-Directional UE.* We construct a backward-leak uni-directional key update and uni-directional ciphertext update UE scheme UNIUE from a key and message homomorphic PKE scheme. The construction is described in Fig. 5. The idea of this construction is that only the public value  $pk_{e+1,e+1}$  is included in the update token  $\Delta_{e+1}$ , secret key  $sk_{e+1,e+1}$  is not included, in other words, no information of this secret key can be revealed from the token. We can deploy this key pair to protect the confidentiality in the new epoch. The detailed construction is shown in Fig. 5. When epoch period turns into the next epoch period, the new epoch key will increase one key element. That is, epoch key  $\mathbf{k}_e$  has  $e$  pairs of secret key and public key, epoch key  $\mathbf{k}_{e+1}$  has  $e + 1$  pairs of secret key and public key. To update a ciphertext, we use the difference of  $\mathbf{sk}_e$  and  $\mathbf{sk}_{e+1}$  (except for the last element) to move encryption of random elements from epoch  $e$  to epoch  $e + 1$ . Due to PKE is message homomorphic, we can perform a re-randomization to refresh the underlying random values. Then, we add an encryption of a new random value, which is encrypted under  $pk_{e+1,e+1}$ , into the updated ciphertext. This new random value can be used to hide the message. Note that if we do not include this additional randomness, the above construction is bi-directional.

*Correctness.* It is sufficient to prove that the updated ciphertext is a ciphertext in epoch  $e+1$ , with underlying message  $m$ . We compute the updated ciphertext as follows.  $\mathbf{c}^1 = \text{PKE.KHC}(\Delta_{e+1}^{sk}, \mathbf{c}_{e,1}) \stackrel{\text{Equation (7)}}{\approx} \text{PKE.Enc}([\mathbf{sk}_e \otimes \Delta_{e+1}^{sk}], \mathbf{R}_e)$  and  $\mathbf{c}_{e+1,1} \leftarrow (\mathbf{c}^1, 0) + \text{PKE.Enc}(\mathbf{pk}_{e+1}, \mathbf{R}) = \text{PKE.Enc}(\mathbf{pk}_{e+1}, (\mathbf{R}_e, 0) + \mathbf{R})$ . Denote  $\mathbf{R}_{e+1} = (r_{e+1,1}, \dots, r_{e+1,e+1}) = (\mathbf{R}_e, 0) + \mathbf{R}$ , due to the randomness of  $\mathbf{R}$  we know  $\mathbf{R}_{e+1}$  is a random vector. Furthermore,  $c_{e+1,2} \leftarrow c_{e,2} \oplus r_1 \oplus \dots \oplus r_{e+1} = r_{e+1,1} \oplus \dots \oplus r_{e+1,e+1} \oplus m$ . Therefore, the updated ciphertext  $\mathbf{c}_{e+1}$  is a valid ciphertext in epoch  $e + 1$ . Note that we consider epoch bounded UE, which implies that the noise in lattice-based constructions will not grow too large.

*Backward-Leak Uni-Directional Key Updates.* Any new key  $\mathbf{sk}_{e+1}$  has a random key element  $sk_{e+1,e+1}$  which is independent from the update token  $\Delta_{e+1}$  and the previous key  $\mathbf{sk}_e$ , hence, any adversary cannot upgrade keys.

*Uni-Directional Ciphertext Updates.* If there exists an adversary  $\mathcal{A}$  which can infer a valid previous ciphertext  $\mathbf{c}_e$  from token  $\Delta_{e+1}$  and ciphertext  $\mathbf{c}_{e+1}$ . Then we claim that  $\mathcal{A}$  can use this ability to win the IND $\$$ -CPA game for PKE. Initially,  $\mathcal{A}$  receives a public key  $pk_{e+1,e+1}$  from its IND $\$$ -CPA challenger.  $\mathcal{A}$  generates a secret key  $\mathbf{sk}_e$  and a token  $\Delta_{e+1}^{sk}$  as algorithms of UNIUE specify.  $\mathcal{A}$  computes the public key  $\mathbf{pk}_{e+1}$  and token  $\Delta_{e+1}$  by embedding the public key  $pk_{e+1,e+1}$ . Suppose  $\mathcal{A}$  asks for a challenge query with input  $r_{e+1,e+1}$ , it gets the challenge ciphertext  $\tilde{c}$ . Then  $\mathcal{A}$  uses  $r_{e+1,e+1}$  and  $\tilde{c}$  to create a ciphertext in epoch  $e + 1$ , say  $\tilde{\mathbf{c}}_{e+1}$ .  $\mathcal{A}$  can move the ciphertext  $\tilde{\mathbf{c}}_{e+1}$  to a ciphertext  $\tilde{\mathbf{c}}_e$  by the token  $\Delta_{e+1}$

<pre> Setup(<math>\lambda</math>) :   <math>(sk_{1,1}, pk_{1,1}) \leftarrow \text{PKE.KG}(\lambda)</math>   return <math>(sk_{1,1}, pk_{1,1})</math>  Next(<math>sk_e</math>) :   parse <math>sk_e = (sk_{e,1}, \dots, sk_{e,e})</math>   for <math>i \in \{1, \dots, e\}</math> do     <math>(\Delta_i, [\Delta_i]) \leftarrow \text{PKE.KG}(\lambda)</math>     <math>sk_{e+1,i} \leftarrow sk_{e,i} \otimes \Delta_i</math>     <math>pk_{e+1,i} \leftarrow [sk_{e+1,i}]</math>   <math>(sk_{e+1,e+1}, pk_{e+1,e+1}) \leftarrow \text{PKE.KG}(\lambda)</math>   <math>sk_{e+1} \leftarrow (sk_{e+1,1}, \dots, sk_{e+1,e+1})</math>   <math>pk_{e+1} \leftarrow (pk_{e+1,1}, \dots, pk_{e+1,e+1})</math>   <math>\Delta_{e+1}^{sk} \leftarrow (\Delta_1, \dots, \Delta_e)</math>   <math>\Delta_{e+1} \leftarrow (\Delta_{e+1}^{sk}, pk_{e+1,e+1})</math>   return <math>\Delta_{e+1}, (sk_{e+1}, pk_{e+1})</math>  Enc(<math>pk_e, m</math>) :   <math>R_e \xleftarrow{\\$} \mathcal{M}^{e \times 1}</math>   parse <math>R_e = (r_{e,1}, \dots, r_{e,e})</math>   <math>c_{e,1} \leftarrow \text{PKE.Enc}(pk_e, R_e)</math>   <math>c_{e,2} \leftarrow r_{e,1} \oplus \dots \oplus r_{e,e} \oplus m</math>   return <math>c_e = (c_{e,1}, c_{e,2})</math> </pre>	<pre> Dec(<math>sk_e, c_e</math>) :   parse <math>c_e = (c_{e,1}, c_{e,2})</math>   <math>R_e \leftarrow \text{PKE.Dec}(sk_e, c_{e,1})</math>   parse <math>R_e = (r_{e,1}, \dots, r_{e,e})</math>   <math>m' \leftarrow c_{e,2} \oplus^{-1} (r_{e,1} \oplus \dots \oplus r_{e,e})</math>   return <math>m'</math>  Upd(<math>\Delta_{e+1}, c_e</math>) :   parse <math>\Delta_{e+1} = (\Delta_{e+1}^{sk}, pk_{e+1})</math>   parse <math>c_e = (c_{e,1}, c_{e,2})</math>   <math>R \xleftarrow{\\$} \mathcal{M}^{(e+1) \times 1}</math>   <math>c^1 \leftarrow \text{PKE.KHC}(\Delta_{e+1}^{sk}, c_{e,1})</math>   <math>c_{e+1,1} \leftarrow (c^1, 0) + \text{PKE.Enc}(pk_{e+1}, R)</math>   parse <math>R = (r_1, \dots, r_{e+1})</math>   <math>c_{e+1,2} \leftarrow c_{e,2} \oplus r_1 \oplus \dots \oplus r_{e+1}</math>   <math>c_{e+1} \leftarrow (c_{e+1,1}, c_{e+1,2})</math>   return <math>c_{e+1}</math> </pre>
--	--

**Fig. 5.** UNIUE = (Setup, Next, Enc, Dec, Upd) is a UE scheme built from a key and message homomorphic PKE scheme PKE.  $\oplus$  is an operation on the message space and assume its inverse operation exists.

and then decrypt  $\tilde{c}_e$  by the secret key  $sk_e$ . Eventually,  $\mathcal{A}$  compares the message with the message used when it creates  $\tilde{c}_{e+1}$ . If they are the same then  $\mathcal{A}$  guesses it received a real encryption from its IND $\$$ -CPA challenger, otherwise, it guesses it received a random ciphertext from its IND $\$$ -CPA challenger. The advantage of  $\mathcal{A}$  winning the IND $\$$ -CPA game is equal to the probability of  $\mathcal{A}$  successfully downgrades the ciphertext  $c_{e+1}$  to a ciphertext  $c_e$  by the token  $\Delta_{e+1}$ .

Next, we prove that the UE scheme constructed in Fig. 5 satisfies the stronger variant of rand-IND-UE security, namely the (b-uni, uni)-rand-IND-UE security.

**Theorem 3.** *Let UNIUE be the updatable encryption scheme described in Fig. 5. For any (b-uni, uni)-rand-IND-UE adversary  $\mathcal{A}$  against UNIUE, there exists an IND $\$$ -CPA adversary  $\mathcal{B}_3$  against PKE such that*

$$\mathbf{Adv}_{\text{UNIUE}, \mathcal{A}}^{(\text{b-uni, uni})\text{-randIND-UE-CPA}}(\lambda) \leq 2l^2 \cdot \mathbf{Adv}_{\text{PKE}, \mathcal{B}_3}^{\text{IND}\$-\text{CPA}} + \text{negl}(\lambda),$$

where  $l$  is the upper bound on the last epoch.

*Remark 2.* The difference between proving a UE scheme is (b-uni, uni)-rand-IND-UE secure and (bi, bi)-rand-IND-UE secure are trivial win conditions, and how the reduction runs the simulation.

Consider the backward-leak uni-directional key updates variant of a confidentiality notion, there will exist a the relaxed insulated region (recall the discussion in Sect. 3.2). Assume  $\tilde{e}$  is the challenge epoch, where key in this epoch should not be corrupted. Hence, there exists an epoch after  $\tilde{e}$ , say  $\text{fwr}$ , such that keys in epoch  $\{\tilde{e}, \dots, \text{fwr}\}$  and token in epoch  $\text{fwr} + 1$  are not corrupted, in addition, tokens in epoch  $\{\tilde{e} + 1, \dots, \text{fwr}\}$  are corrupted. By Definition 4, we know that epoch region  $\{\tilde{e}, \dots, \text{fwr}\}$  is a relaxed insulated region. Tokens and keys before  $\tilde{e}$  can be corrupted, which will not trigger the trivial win condition. Because knowing any token and any key before epoch  $\tilde{e}$  will not break the confidentiality in epoch  $\tilde{e}$ , the key element  $sk_{\tilde{e}, \tilde{e}}$  in  $\mathbf{sk}_{\tilde{e}}$  is an independent and random value compared to all previous update tokens and keys, hence, ciphertexts in epoch  $\tilde{e}$  are random looking without the knowledge of the key element  $sk_{\tilde{e}, \tilde{e}}$ .

*Proof.* The proof is similar to the proof in Theorem 2. We construct hybrid games and apply the firewall technique on relaxed insulated regions.

*Step 1.* In the initial hybrid games, we move challenges from real to random over epochs. We construct a sequence of hybrid games  $H_1, \dots, H_l$ . For  $b \in \{0, 1\}$ , experiment  $H_i^b$  is defined as follows, if the adversary asks for a challenge-equal ciphertext by the  $\mathcal{O}.\text{Chall}$  query or a  $\mathcal{O}.\text{Upd}\tilde{\mathcal{C}}$  query, with challenge input  $(\tilde{m}, \tilde{c})$ , in epoch  $j$ :

- if  $j \leq i$ , for  $b = 1$  return an updated ciphertext of  $\tilde{c}$ , for  $b = 0$  return (an updated ciphertext which is updated from) an encrypted ciphertext of  $\tilde{m}$ .
- if  $j > i$ , return a random ciphertext.

Thus  $H_l^1$  is  $\mathbf{Exp}_{\text{UNIUE}, \mathcal{A}}^{(b\text{-uni, uni})\text{-randIND-UE-CPA-1}}$ , i.e. all challenge responses are challenge-equal ciphertexts of  $\text{Upd}(\tilde{c})$ . And  $H_l^0$  is  $\mathbf{Exp}_{\text{UNIUE}, \mathcal{A}}^{(b\text{-uni, uni})\text{-randIND-UE-CPA-0}}$ , i.e. all challenge responses are challenge-equal ciphertexts of  $\text{Enc}(\tilde{m})$ . Notice that  $H_0^0 = H_0^1$ , in which all challenge ciphertexts are random ciphertexts. We have

$$\begin{aligned} \mathbf{Adv}_{\text{UNIUE}, \mathcal{A}}^{(b\text{-uni, uni})\text{-randIND-UE-CPA}} &= |\Pr[H_l^1 = 1] - \Pr[H_l^0 = 1]| \\ &\leq \sum_{i=1}^l |\Pr[H_i^1 = 1] - \Pr[H_{i-1}^1 = 1]| \\ &\quad + \sum_{i=1}^l |\Pr[H_i^0 = 1] - \Pr[H_{i-1}^0 = 1]|. \end{aligned}$$

*Step 2.* We define a new game  $\mathcal{G}_i$  that is the same as game  $H_i$ , except for the game randomly picks a number  $\text{fwr} \stackrel{\$}{\leftarrow} \{0, \dots, l\}$ . If the adversary corrupts a key in the sequence of epochs  $(i, \dots, \text{fwr})$  or a token in epoch  $\text{fwr} + 1$ , the game aborts. This loss is upper bounded by  $l$ .

Then we have  $|\Pr[H_i^b = 1] - \Pr[H_{i-1}^b = 1]| \leq l|\Pr[\mathcal{G}_i^b = 1] - \Pr[\mathcal{G}_{i-1}^b = 1]|$ .

*Step 3.* In this step, we prove that  $|\Pr[\mathcal{G}_i^b = 1] - \Pr[\mathcal{G}_{i-1}^b = 1]| = \text{Adv}_{\text{PKE}, \mathcal{B}_3}^{\text{IND\$-CPA}} + \text{negl}(\lambda)$ . Assume  $\mathcal{A}_i$  is an adversary attempting to distinguish  $\mathcal{G}_i^b$  from  $\mathcal{G}_{i-1}^b$ . We construct a reduction  $\mathcal{B}_3$ , detailed in Fig. 6, playing the IND\\$-CPA game by simulating the responses to adversary  $\mathcal{A}_i$ .

Initially, the reduction guesses a number  $\text{fwr}$ . If  $\mathcal{A}_i$  corrupts  $\mathbf{k}_i, \dots, \mathbf{k}_{\text{fwr}}$ , or  $\Delta_{\text{fwr}+1}$  the reduction aborts the game.

A summary of the technical simulations are as follows.

- In the setup phrase,  $\mathcal{B}_3$  generates all keys and tokens, except for  $\mathbf{k}_i, \dots, \mathbf{k}_{\text{fwr}}$ ,  $\Delta_{\text{fwr}+1}$ , as follows.
  - The key pairs and tokens outside of the relaxed insulated regions are generated as in  $\mathcal{G}_i$ .
  - The public keys within relaxed firewalls are generated by embedding public key  $pk$  to the  $i$ -th term of  $\mathbf{pk}_i$ , where  $pk$  is the public key received from its IND\\$-CPA game.
- To simulate non-challenge ciphertexts:  $\mathcal{B}_3$  uses public keys to simulate encrypted ciphertexts and updated ciphertexts.
- To simulate challenge-equal ciphertexts in an epoch that is:
  - $j < i$ :  $\mathcal{B}_3$  uses public keys to simulate encryption and updating.
  - $j = i$ :  $\mathcal{B}_3$  embeds the challenge ciphertext  $\tilde{c}$  received from its IND\\$-CPA challenger to the challenge-equal ciphertext in epoch  $i$ . More precisely, suppose  $\mathcal{B}_3$  receives a challenge query  $\mathcal{O}.\text{Chall}$  with input  $(\tilde{m}_0, \tilde{c})$  in challenge epoch  $\tilde{e}$ , where the underlying message of  $\tilde{c}$  is  $\tilde{m}_1$ .  $\mathcal{B}_3$  sends a random value  $r_i$  to its IND\\$-CPA challenger and obtains  $\tilde{c}_\beta$ .  $\mathcal{B}_3$  embeds  $\tilde{c}_\beta$  to the  $i$ -th term of the challenge ciphertext and uses  $r_i$  and  $m_b$  to compute the last term of the challenge ciphertext. Afterwards,  $\mathcal{B}_3$  returns  $\tilde{c}_i$  to the adversary  $\mathcal{A}$ . Again, by Eq. (7),  $\mathcal{B}_3$  perfectly simulate the challenge ciphertexts in game  $\mathcal{G}_{i-1+\beta}$  except for a negligible probability  $\text{negl}(\lambda)$ .
  - $j > i$ :  $\mathcal{B}_3$  outputs random ciphertext as challenge ciphertext.

Eventually,  $\mathcal{B}_3$  receives the output bit from  $\mathcal{A}_i$  and if  $\mathcal{A}_i$  guesses it is playing  $\mathcal{G}_i$  (suppose it represents the guess response of  $\mathcal{A}_i$  is 1), then  $\mathcal{B}_3$  guesses it received a real encryption and sends 1 to its IND\\$-CPA challenger. Otherwise, sends 0 to its IND\\$-CPA challenger. We have  $|\Pr[\mathcal{G}_i^b = 1] - \Pr[\mathcal{G}_{i-1}^b = 1]| = \text{Adv}_{\text{PKE}}^{\text{IND\$-CPA}} + \text{negl}(\lambda)$ .

## 6 UE from Bootstrappable PKE

Bootstrappability can be used to refresh ciphertexts without revealing the underlying message, which implies updatable encryption.

**Definition 7 (Bootstrappable PKE).** *We say a public key encryption  $\text{BPKE} = (\text{KG}, \text{Enc}, \text{Dec})$  is bootstrappable if it can evaluate its own decryption circuit  $D$ . More precisely, there exists a re-encryption algorithm  $\text{ReCrypt}$  that takes a public key, the decryption circuit  $D$ , an encryption of a secret key and a ciphertext as input and outputs a new ciphertext, such that for any keys*

For  $b \in \{0, 1\}$   $\mathcal{B}_3$  plays  
 $\text{IND\$-CPA}$  game by running  $\mathcal{A}_i$  :

```

receive  $pk$ 
do Setup
 $b' \leftarrow \mathcal{A}_i^{\text{oracles}}(\lambda)$ 
if ABORT occurred or  $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ 
  or  $i > \text{fwr}$  then
     $b' \stackrel{\$}{\leftarrow} \{0, 1\}$ 
return  $b'$ 
    
```

**Setup**( $\lambda$ )

```

 $\Delta_1 \leftarrow \perp$ ;  $e \leftarrow 1$ ;  $\text{phase}, \text{twf} \leftarrow 0$ ;
 $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$ 
 $\text{fwr} \stackrel{\$}{\leftarrow} \{0, \dots, l\}$ 
 $(sk_{1,1}, pk_{1,1}) \leftarrow \text{PKE.KG}(\lambda)$ 
for  $j \in \{2, \dots, i-1\}$  do
   $\Delta_j, (sk_j, pk_j) \leftarrow \text{UNIUE.Next}(sk_{j-1})$ 

for  $j \in \{i, \dots, \text{fwr}\}$  do
  parse  $pk_{j-1} = (pk_{j-1,1}, \dots, pk_{j-1,j-1})$ 
  for  $t \in \{1, \dots, j-1\}$  do
     $(\Delta_t, \cdot) \leftarrow \text{PKE.KG}(\lambda)$ 
     $pk_{j,t} \leftarrow \text{KHK}(\Delta_t, pk_{j-1,t})$ 
    if  $j = i$  then
       $pk_{i,i} \leftarrow pk$ 
    else
       $(\cdot, pk_{j,j}) \leftarrow \text{PKE.KG}(\lambda)$ 
   $\Delta_j^{sk} \leftarrow (\Delta_1, \dots, \Delta_{j-1})$ 
   $pk_j = (pk_{j,1}, \dots, pk_{j,j})$ 
for  $j = \text{fwr}+1$  do
  for  $t \in \{1, \dots, \text{fwr}+1\}$  do
     $(sk_{\text{fwr}+1,t}, pk_{\text{fwr}+1,t}) \leftarrow \text{PKE.KG}(\lambda)$ 
   $sk_{\text{fwr}+1} = (sk_{\text{fwr}+1,1}, \dots, sk_{\text{fwr}+1,\text{fwr}+1})$ 
   $pk_{\text{fwr}+1} = (pk_{\text{fwr}+1,1}, \dots, pk_{\text{fwr}+1,\text{fwr}+1})$ 
for  $j \in \{\text{fwr}+2, \dots, l\}$  do
   $\Delta_j, (sk_j, pk_j) \leftarrow \text{UNIUE.Next}(sk_{j-1})$ 
    
```

$\mathcal{O}.\text{Enc}(m)$  :

```

 $c_e \leftarrow \text{UNIUE.Enc}(pk_e, m)$ 
 $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, c_e, e; m)\}$ 
return  $c_e$ 
    
```

$\mathcal{O}.\text{Next}$  :

```

 $e \leftarrow e+1$ 
    
```

$\mathcal{O}.\text{Upd}(c_{e-1})$  :

```

if  $(\cdot, c_{e-1}, e-1; m) \notin \mathcal{L}$  then
  return  $\perp$ 
    
```

```

 $c_e \leftarrow \text{UNIUE.Enc}(pk_e, m)$ 
 $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, c_e, e; m)\}$ 
return  $c_e$ 
    
```

$\mathcal{O}.\text{Corr}(\text{inp}, \hat{e})$  :

```

if  $(\text{inp} = \text{key}$  and  $\hat{e} \in \{i, \dots, \text{fwr}\})$ 
or  $(\text{inp} = \text{token}$  and  $\hat{e} = \text{fwr}+1)$  then
  ABORT
else
  do as  $\mathcal{O}.\text{Corr}(\text{inp}, \hat{e})$  specifies
    
```

$\mathcal{O}.\text{Chall}(\bar{m}_0, \bar{c})$  :

```

if  $\text{phase} = 1$  then
  return  $\perp$ 
 $\text{phase} \leftarrow 1$ ;  $\hat{e} \leftarrow e$ 
if  $(\cdot, \bar{c}, \hat{e}-1; \bar{m}_1) \notin \mathcal{L}$  then
  return  $\perp$ 
for  $j \in \{1, \dots, i-1\}$  do
   $\tilde{c}_j \leftarrow \text{UNIUE.Enc}(pk_j, m_b)$ 
for  $j = i$  do
   $r_i \stackrel{\$}{\leftarrow} \mathcal{M}$ 
  Send  $r_i$  to the IND\$-CPA challenger,
  get  $\tilde{c}_\beta$ 
  parse  $pk_i = (pk_{i,1}, \dots, pk_{i,i-1})^\top$ 
   $(\tilde{c}'_{i,1}, c_{i,2}) \leftarrow \text{UNIUE.Enc}(pk_i, m_b)$ 
   $\tilde{c}_{i,1} \leftarrow (\tilde{c}'_{i,1}, \tilde{c}_\beta)$ 
   $\tilde{c}_{i,2} \leftarrow c_{i,2} \oplus r_i$ 
   $\tilde{c}_i \leftarrow (\tilde{c}_{i,1}, \tilde{c}_{i,2})$ 
for  $j \in \{i+1, \dots, l\}$  do
   $\tilde{c}_j \stackrel{\$}{\leftarrow} \mathcal{CS}$ 
   $\mathcal{C} \leftarrow \mathcal{C} \cup \{j\}$ 
   $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{c}_j, j)\}$ 
return  $\tilde{c}_e$ 
    
```

$\mathcal{O}.\text{Upd}\tilde{\mathcal{C}}$  :

```

if  $\text{phase} \neq 1$  then
  return  $\perp$ 
 $\mathcal{C} \leftarrow \mathcal{C} \cup \{e\}$ 
 $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{c}_e, e)\}$ 
return  $\tilde{c}_e$ 
    
```

**Fig. 6.** Reduction  $\mathcal{B}_3$  for proof of Theorem 3.

$(sk_1, pk_1), (sk_2, pk_2) \leftarrow \text{KG}(\lambda)$  and any message  $m$ , the following two distributions are statistically close:

$$(c, \text{Recrypt}(pk_2, D, \text{Enc}(pk_2, sk_1), c)) \stackrel{s}{\approx} (c, \text{Enc}(pk_2, m)), \tag{8}$$

where  $c = \text{Enc}(pk_1, m)$ .

Note that bootstrappable PKE is simpler than a FHE scheme. Most FHE scheme requires bootstrappability, while only bootstrappability is not enough for FHE. Gentry [8, Chapter 4] constructed a re-encryption algorithm `Recrypt` (see Fig. 7), which allows us to update a ciphertext under  $pk_1$  to a ciphertext under  $pk_2$ .

```

Recrypt( $pk_2, D, \langle \overline{sk_{1,j}} \rangle, c_1$ ) :
   $\overline{c_{1,j}} \xleftarrow{s} \text{BPKE.Enc}(pk_2, c_{1,j})$ 
   $c_2 \leftarrow \text{BPKE.Evaluate}(pk_2, D, \langle \langle \overline{sk_{1,j}} \rangle, \langle \overline{c_{1,j}} \rangle \rangle)$ 
  return  $c_2$ 
    
```

**Fig. 7.** `Recrypt` algorithm. For any key pairs  $(sk_1, pk_1), (sk_2, pk_2) \leftarrow \text{KG}(\lambda)$ . Let  $sk_{1,j}$  be the  $j$ -th bit of  $sk_1$  and  $\overline{sk_{1,j}} = \text{Enc}(pk_2, sk_{1,j})$ . For any plaintext  $m \in \mathcal{M}$ , let  $c_1 = \text{Enc}(pk_1, m)$ , and  $c_{1,j}$  denote the  $j$ -th bit of  $c_1$ . The output  $c_2$  is an encryption of  $\text{Dec}(sk_1, c_1) = m$  under  $pk_2$ .

The `Recrypt` algorithm can be used to update ciphertext, where the update token is the encryption of the current secret key  $sk_e$  under the next public key  $pk_{e+1}$ . We construct an updatable encryption scheme `BPKEUE` from `BPKE`, which is shown in Fig. 8.

<pre> Setup(<math>\lambda</math>) :   <math>(sk_1, pk_1) \leftarrow \text{BPKE.KG}(\lambda)</math>   return <math>(sk_1, pk_1)</math>  Next(<math>sk_e</math>) :   <math>(sk_{e+1}, pk_{e+1}) \leftarrow \text{BPKE.KG}(\lambda)</math>   <math>\Delta_{e+1} \leftarrow \text{BPKE.Enc}(pk_{e+1}, sk_e)</math>   return <math>\Delta_{e+1}, (sk_{e+1}, pk_{e+1})</math>         </pre>	<pre> Enc(<math>pk_e, m</math>) :   <math>c_e \leftarrow \text{BPKE.Enc}(pk_e, m)</math>   return <math>c_e</math>  Dec(<math>sk_e, c_e</math>) :   <math>m' \leftarrow \text{BPKE.Dec}(sk_e, c_e)</math>   return <math>m'</math>  Upd(<math>\Delta_{e+1}, c_e</math>) :   <math>c_{e+1} \leftarrow \text{BPKE.Recrypt}(pk_{e+1}, D, \Delta_{e+1}, c_e)</math>   return <math>c_{e+1}</math>         </pre>
--	--

**Fig. 8.** `BPKEUE = (Setup, Next, Enc, Dec, Upd)` is a UE scheme constructed from a bootstrappable PKE scheme `BPKE`.

*Correctness.* The correctness of encrypting then decrypting follows the correctness of the underlying PKE scheme. The correctness of encrypting then updating then decrypting is because of the bootstrappability of `BPKE` scheme, the

re-encrypted ciphertext is a new ciphertext encrypted under the new public key with the same message. Note that we consider epoch bounded UE, which implies that the noise will not grow too large.

*Backward-Leak Uni-Directional Key Updates.* We can see the earlier key  $sk_e$  and token  $\Delta_{e+1}$  as a plaintext and the corresponding ciphertext under public key  $pk_{e+1}$ . Hence, any adversary can not obtain the secret key  $sk_{e+1}$ .

*Uni-Directional Ciphertext Updates.* If there exists an adversary which can infer a valid previous ciphertext  $c_e$  from token  $\Delta_{e+1}$  and ciphertext  $c_{e+1}$ . Then we claim that the adversary can use this ability to win the IND-CPA game for BPKE. Initially, the adversary receives a public key  $pk_{e+1}$  from its IND-CPA challenger. The adversary generates a secret key  $sk_e$  and computes the token  $\Delta_{e+1}$  by the knowledge of the public key  $pk_{e+1}$  and the secret key  $sk_e$ . Then the adversary can move the challenge ciphertext  $\tilde{c}_{e+1}$  (encrypted under  $pk_{e+1}$ ) to a ciphertext  $\tilde{c}_e$  by the token  $\Delta_{e+1}$ . Note that  $\tilde{c}_{e+1}$  and  $\tilde{c}_e$  have the same underlying message. Therefore, the adversary can decrypt  $\tilde{c}_e$  by  $sk_e$  and then compare the output with the challenge messages to win the IND-CPA game.

*Remark 3.* Nishimaki [14] observed that if the update token  $\Delta_{e+1}$  is generated by the old secret key  $sk_e$  and the new public key  $pk_{e+1}$ , such UE schemes may have backward-leak uni-directional key updates. The reason is that it will be difficult to break the confidentiality in epoch  $e + 1$  with only the knowledge of  $pk_{e+1}$ ,  $\Delta_{e+1}$  and  $sk_e$ , no information about  $sk_{e+1}$  is revealed.

We observed that such UE schemes may have uni-directional ciphertext updates as well. The proof idea is similar to the proof of BPKEUE has uni-directional ciphertext updates. We claim that if such UE schemes do not have uni-directional ciphertext updates, then any adversary can break the confidentiality of such UE schemes without the knowledge of any epoch key. If an adversary aims to attack the confidentiality in epoch  $e + 1$ , it can generate a new secret key in epoch  $e$ . Then the adversary computes the token  $\Delta_{e+1}$  by the generated secret key  $sk_e$  and the public key  $pk_{e+1}$ . It can move ciphertexts from epoch  $e + 1$  to epoch  $e$  by token  $\Delta_{e+1}$  and then decrypt it by  $sk_e$  to win the confidentiality in epoch  $e + 1$ .

Next, we prove that the UE scheme constructed in Fig. 8 is secure under the (b-uni, uni)-rand-IND-UE notion.

**Theorem 4.** *Let BPKEUE be the UE scheme described in Fig. 8. For any (b-uni, uni)-rand-IND-UE adversary  $\mathcal{A}$  against BPKEUE, there exists an IND-CPA adversary  $\mathcal{B}_4$  against BPKE such that*

$$\text{Adv}_{\text{BPKEUE}, \mathcal{A}}^{(\text{b-uni, uni})\text{-randIND-UE-CPA}}(\lambda) \leq 2l^3 \cdot \text{Adv}_{\text{BPKE}, \mathcal{B}_4}^{\text{IND-CPA}} + \text{negl}(l),$$

where  $l$  is the upper bound on the last epoch.

Before proving the above theorem, we prove a lemma first. In the IND game, any adversary can only ask for tokens and encryption oracles.

**Lemma 5.** *Let BPKEUE be the UE scheme described in Fig. 8. For any IND adversary  $\mathcal{A}$  against BPKEUE, there exists an IND-CPA adversary  $\mathcal{B}_5$  against BPKE such that*

$$\text{Adv}_{\text{BPKEUE}, \mathcal{A}}^{\text{IND}}(\lambda) \leq 2l \cdot \text{Adv}_{\text{BPKE}, \mathcal{B}_5}^{\text{IND-CPA}},$$

where  $l$  is the upper bound on the last epoch.

*Proof (of Lemma 5).* The proof is similar to the proof of Theorem 4.2.3 in [8]. We use a hybrid games to move tokens from real to random. In hybrid  $i$ , the last  $l - i$  tokens are random from the real keys. More precisely, for  $i \in \{1, \dots, l\}$  let  $\mathcal{G}_i$  be a game that is identical to the IND game against BPKEUE, except for all  $j > i$ :

$$(sk'_j, pk'_j) \stackrel{\$}{\leftarrow} \text{KG}(\lambda), \Delta_j \stackrel{\$}{\leftarrow} \text{Enc}(pk_j, sk'_{j-1}).$$

Note that the last  $l - i$  tokens are not related to the real keys.

We have that  $\mathcal{G}_l$  is the IND game and the advantage of any adversary winning  $\mathcal{G}_0$  is upper bounded by  $l \cdot \text{Adv}_{\text{BPKE}}^{\text{IND-CPA}}$ . Next, we claim that for any  $i \in \{1, \dots, l\}$ ,  $|\Pr[\mathcal{G}_i = 1] - \Pr[\mathcal{G}_{i-1} = 1]| = \text{Adv}_{\text{BPKE}}^{\text{IND-CPA}}$ .

Suppose  $\mathcal{A}$  is an adversary aiming to distinguish  $\mathcal{G}_i$  from  $\mathcal{G}_{i-1}$ . We construct a reduction  $\mathcal{B}_5$  playing the IND-CPA game (against BPKE) and simulating the response to  $\mathcal{A}$ . Initially,  $\mathcal{B}_5$  receives a public key  $pk$  from its IND-CPA challenger.  $\mathcal{B}_5$  generates key pairs as in  $\mathcal{G}_i$  except for it embeds  $pk$  to  $pk_i$ . It generates a random key pair  $(sk'_{i-1}, pk'_{i-1}) \stackrel{\$}{\leftarrow} \text{KG}(\lambda)$ , sets  $(m_0, m_1) = (sk'_{i-1}, sk_{i-1})$  and sends  $(m_0, m_1)$  to its challenger. The challenger flips a coin  $\beta \stackrel{\$}{\leftarrow} \{0, 1\}$  and returns the encryption of  $m_\beta$ .  $\mathcal{B}_5$  sets the received challenge ciphertext as  $\Delta_j$ . Note that  $\mathcal{B}_5$  perfectly simulates public keys and tokens in  $\mathcal{G}_{i-1+\beta}$  to  $\mathcal{A}$ . When  $\mathcal{A}$  asks for a challenge query on  $(\bar{m}_0, \bar{m}_1)$ ,  $\mathcal{B}_5$  flips a coin  $b \stackrel{\$}{\leftarrow} \{0, 1\}$  and sends the encryptions of  $\bar{m}_b$  to  $\mathcal{A}$ . Eventually,  $\mathcal{A}$  submit a guess, if  $\mathcal{A}$  guesses it is  $\mathcal{G}_i$  then  $\mathcal{B}_5$  returns 1 to its challenger, otherwise,  $\mathcal{B}_5$  returns 0.

Since  $\mathcal{B}_5$  perfectly simulate  $\mathcal{G}_{i-1+\beta}$  to  $\mathcal{A}$ . The probability of  $\mathcal{A}$  is able to distinguish which game it is playing is equal to  $\text{Adv}_{\text{BPKE}, \mathcal{B}_5}^{\text{IND-CPA}}$ .

*Proof (of Theorem 4).* We use firewall technique and construct a sequence of hybrid games to move challenges from left to right over the relaxed insulated regions. Define game  $\mathcal{G}_i$  as (b-uni, uni)-randIND-UE-CPA game, except for

- The game randomly choose a number  $\text{fwr} \stackrel{\$}{\leftarrow} \{0, \dots, l\}$ . If  $\text{fwr}$  is not the  $i$ -th right firewall, returns a random bit for  $b'$ . This loss is upper bounded by  $l$ .
- To the left side of epoch  $\text{fwr}$ , the game returns a ciphertext with respect to  $\bar{c}$ , to the right side of epoch  $\text{fwr}$  returns a encryption of  $\bar{m}$ .

If  $\text{fwr}$  is guessed correct, then  $\mathcal{G}_0$  is  $\text{Exp}_{\text{BPKEUE}}^{(\text{b-uni, uni})\text{-randIND-UE-CPA-0}}$  and  $\mathcal{G}_l$  is  $\text{Exp}_{\text{BPKEUE}}^{(\text{b-uni, uni})\text{-randIND-UE-CPA-1}}$ . So we can bound the (b-uni, uni)-randIND-UE-CPA advantage by the advantage of distinguishing  $\mathcal{G}_0$  and  $\mathcal{G}_l$ .

$$\text{Adv}_{\text{BPKEUE}, \mathcal{A}}^{(\text{b-uni, uni})\text{-randIND-UE-CPA}}(\lambda) \leq \sum_{i=1}^l |\Pr[\mathcal{G}_i = 1] - \Pr[\mathcal{G}_{i-1} = 1]|,$$



Notice that if  $\mathcal{G}_{i-1}$  and  $\mathcal{G}_i$  have the same right firewall  $\text{fwr}$ , then they are the same game, hence, we assume  $\mathcal{G}_{i-1}$  and  $\mathcal{G}_i$  have different right firewall. Suppose  $\mathcal{A}_i$  is an adversary attempting to distinguish  $\mathcal{G}_{i-1}$  from  $\mathcal{G}_i$ . For all queries concerning epochs outside of the  $i$ -th relaxed insulated region ( $\{i, \dots, \text{fwr}\}$ ) the responses will be equal in either game. We construct a reduction  $\mathcal{B}_4$  playing the IND game (within the epoch region  $\{i, \dots, \text{fwr}\}$ ) for BPKEUE and will simulate the responses of queries made by  $\mathcal{A}_i$ . Initially, the reduction guesses a number  $\text{fwr}$ . If  $\mathcal{A}_i$  corrupts  $k_i, \dots, k_{\text{fwr}}$ , or  $\Delta_{\text{fwr}+1}$  the reduction aborts the game. A summary of the technical simulations are as follows.

- In the setup phrase,  $\mathcal{B}_4$  generates all keys and tokens, except for  $k_i, \dots, k_{\text{fwr}}, \Delta_{\text{fwr}+1}$ , as follows.
  - The key pairs and tokens outside of the relaxed insulated regions are generated as in  $\mathcal{G}_i$ .
  - The public key within firewalls are generated by embedding public keys  $(pk_i, \dots, pk_{\text{fwr}})$  and tokens  $(\Delta_{i+1}, \dots, \Delta_{\text{fwr}})$ , which are received from the IND challenger.
- To simulate non-challenge ciphertexts:  $\mathcal{B}_4$  uses public keys to simulate encrypted ciphertexts and updated ciphertexts. Due to updated ciphertext is statistically close to the fresh encryption of the same underlying message, the adversary notice this change with negligible probability.
- To simulate challenge-equal ciphertexts in an epoch that is:
  - $j < i$  or  $j > \text{fwr}$ :  $\mathcal{B}_4$  uses public keys to simulate encryption and updating.
  - $j \in \{i, \dots, \text{fwr}\}$ :  $\mathcal{B}_4$  sends  $(\bar{m}_0, \bar{m}_1)$  to its IND challenger and forwards the response to  $\mathcal{A}_i$ , where  $\bar{m}_1$  is the underlying message of  $\bar{c}$ .

Eventually,  $\mathcal{A}_i$  sends a guess. If  $\mathcal{A}_i$  guesses it is playing  $\mathcal{G}_{i-1}$  then  $\mathcal{B}_4$  guesses 0 to its IND challenger. Otherwise,  $\mathcal{B}_4$  sends 1 to its IND challenger. Note that  $\mathcal{B}_4$  perfectly simulates  $\mathcal{G}_{i-1}$  to  $\mathcal{A}_i$  when its challenger encrypts  $\bar{m}_0$  and perfectly simulates  $\mathcal{G}_i$  to  $\mathcal{A}_i$  when its challenger encrypts  $\bar{m}_1$ .

$$\text{Adv}_{\text{BPKEUE}, \mathcal{A}_i}^{(\text{b-uni, uni})\text{-randIND-UE-CPA}}(\lambda) \leq l^2 \cdot \text{Adv}_{\text{BPKEUE}, \mathcal{B}_4}^{\text{IND}} + \text{negl}(l),$$

Combing the result of Lemma 5, we have the desired result.

**Acknowledgements.** We thank the anonymous reviewers of Eurocrypt 2022, Crypto 2022, and PKC 2023 for their useful comments. We also thank Christoph Striecks and Daniel Slamanig for their valuable suggestions to improve the previous version of our paper.

## References

1. Alamati, N., Montgomery, H., Patranabis, S.: Symmetric primitives with structured secrets. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 650–679. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26948-7\\_23](https://doi.org/10.1007/978-3-030-26948-7_23)

2. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 559–589. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_19](https://doi.org/10.1007/978-3-030-64840-4_19)
3. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40041-4\\_23](https://doi.org/10.1007/978-3-642-40041-4_23)
4. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and Secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 464–493. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-56784-2\\_16](https://doi.org/10.1007/978-3-030-56784-2_16)
5. Chen, L., Li, Y., Tang, Q.: CCA updatable encryption against malicious re-encryption attacks. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 590–620. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_20](https://doi.org/10.1007/978-3-030-64840-4_20)
6. Everspaugh, A., Paterson, K., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 98–129. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_4](https://doi.org/10.1007/978-3-319-63697-9_4)
7. Galteland, Y.J., Pan, J.: Backward-leak UNI-directional updatable encryption from (homomorphic) public key encryption. Cryptology ePrint Archive, Paper 2022/324 (2022). <https://eprint.iacr.org/2022/324>
8. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford, CA, USA (2009)
9. Jiang, Y.: The direction of updatable encryption does not matter much. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 529–558. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64840-4\\_18](https://doi.org/10.1007/978-3-030-64840-4_18)
10. Jiang, Y.: The direction of updatable encryption does not matter much. Cryptology ePrint Archive, Report 2020/622 (2020). <https://ia.cr/2020/622>
11. Kloof, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 68–99. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17653-2\\_3](https://doi.org/10.1007/978-3-030-17653-2_3)
12. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 685–716. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_22](https://doi.org/10.1007/978-3-319-78372-7_22)
13. Miao, P., Patranabis, S., Watson, G.: Unidirectional updatable encryption and proxy re-encryption from DDH or LWE. Cryptology ePrint Archive, Report 2022/311 (2022). <https://ia.cr/2022/311>
14. Nishimaki, R.: The direction of updatable encryption does matter. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022. LNCS, vol. 13178, pp. 194–224. Springer, Cham (2022). [https://doi.org/10.1007/978-3-030-97131-1\\_7](https://doi.org/10.1007/978-3-030-97131-1_7)
15. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 2005 Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM (2005). <https://doi.org/10.1145/1060590.1060603>
16. Slamanig, D., Striecks, C.: Puncture 'em all: Stronger updatable encryption with no-directional key updates. IACR Cryptol. ePrint Arch. 268 (2021). <https://eprint.iacr.org/2021/268>



# Functional Encryption Against Probabilistic Queries: Definition, Construction and Applications

Geng Wang, Shi-Feng Sun, Zhedong Wang, and Dawu Gu<sup>(✉)</sup>

School of Electronic Information and Electrical Engineering,  
Shanghai Jiao Tong University, 200240 Shanghai, P. R. China  
{wanggxx, shifeng.sun, wzdstill, dwgu}@sjtu.edu.cn

**Abstract.** Functional encryption (FE for short) can be used to calculate a function output of a message, without revealing other information about the message. There are mainly two types of security definitions for FE, exactly simulation-based security (SIM-security) and indistinguishability-based security (IND-security). Both of them have some limitations: FE with SIM-security supporting all circuits cannot be constructed for unbounded number of ciphertext and/or key queries, while IND-security is sometimes not enough: there are examples where an FE scheme is IND-secure but not intuitively secure. In this paper, we present a new security definition which can avoid the drawbacks of both SIM-security and IND-security, called indistinguishability-based security against probabilistic queries (pIND-security for short), and we give an FE construction for all circuits which is secure for unbounded key/ciphertext queries under this new security definition. We prove that this new security definition is strictly between SIM-security and IND-security, and provide new applications for FE which were not known to be constructed from IND-secure or SIM-secure FE.

**Keywords:** functional encryption · probabilistic queries · indistinguishability-based security · provable security

## 1 Introduction

Functional encryption (FE) was first introduced by Boneh et al. in 2011 [BSW11], which can calculate the function output  $f(m)$  given the encrypted message  $\text{Enc}(m)$ , and leaks nothing else about the message  $m$ . Functional encryption is a mighty cryptographic primitive, and can be considered as a generalization of attribute-based encryption, predicate encryption and inner product encryption.

Functional encryption is also an important method for computing on encrypted data, especially for cloud computing [KLM+18, RSG+19, MSH+19]. Using functional encryption, the cloud server can take ciphertexts as input, and outputs the required computation result as plaintext. This is different from homomorphic encryption, where the result is a ciphertext that requires additional decryption procedure and may not be suitable for some applications.

Informally, a functional encryption scheme consists of four algorithms: despite the normally defined algorithms **Setup**, **Enc**, **Dec** as in public key encryption, there is another algorithm **KeyGen** in functional encryption, which takes the master secret key and a function  $f \in \mathcal{F}$  as input, and outputs a function key  $sk_f$ . In the decryption algorithm, function key  $sk_f$  instead of the master secret key is used, and the function value  $f(m)$  instead of the message  $m$  itself is returned. (See Sect. 2 for the formal definition.)

There are mainly two types of security definitions for functional encryption: indistinguishability-based security (IND-security) and simulation-based security (SIM-security). However, both of them have their own drawbacks. We first briefly introduce the two types of security notions, then show why it is necessary to define a new type of security notions between them.

### 1.1 Overview of Security Notions for FE

The standard IND-security is equivalent to the natural notion of semantic security in public key encryption, and is also defined for many other cryptographic primitives, such as identity-based and attribute-based encryption. But for functional encryption, it has been pointed out that IND-security is not the strongest security definition. We first informally recall the definition of IND-security for FE:

An adversary  $\mathcal{A}$  cannot distinguish between a ciphertext for  $m_0$  and a ciphertext for  $m_1$ , even if allowed to query secret keys  $\{sk_f\}$  for polynomial many different functions  $\{f \in \mathcal{F}\}$ , providing that  $f(m_0) = f(m_1)$ . (We say that  $\mathcal{A}$  is “admissible” if it only makes queries such that  $f(m_0) = f(m_1)$ .)

It seems to be natural for the restriction  $f(m_0) = f(m_1)$ , since  $\mathcal{A}$  can trivially determine whether the ciphertext is for  $m_0$  or  $m_1$  otherwise. However, such a restriction leads to the counter-intuitive example given in [O’N10,BSW11] and refined in [AGVW13]:

*Example 1.1* ([BSW11,AGVW13]). Let  $\mathcal{F}$  be a family of one-way permutations. Suppose that  $\text{PKE} = (\text{PKE.KeyGen}, \text{PKE.Enc}, \text{PKE.Dec})$  is a secure public-key encryption scheme. Then the following FE construction for  $\mathcal{F}$  is IND-secure:

- **Setup**( $1^\lambda$ ): Let  $(\text{PKE.pk}, \text{PKE.sk}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ , and return  $\text{PK} = \text{PKE.pk}$ ,  $\text{MSK} = \text{PKE.sk}$ .
- **Enc**( $\text{PK}, m$ ): Return  $\text{PKE.Enc}(\text{PK}, m)$ .
- **KeyGen**( $\text{MSK}, f$ ): Return  $(\text{MSK}, f)$ .
- **Dec**( $sk_f, ct_m$ ): Let  $sk_f = (\text{MSK}, f)$ , return  $f(\text{PKE.Dec}(\text{MSK}, ct_m))$ .

However, each  $sk_f$  totally leaks  $m$ , while  $f(m)$  does not leak  $m$  (since  $f$  is one-way).

It is not difficult to understand why such a counter-example exists: since the adversary is only allowed to query on  $f$  such that  $f(m_0) = f(m_1)$ , it is not allowed to make any single key query if  $\mathcal{F}$  is a family of one-way permutations.

In [BSW11], the authors defined a stronger security notion, called simulation-based security to handle such cases. Informally speaking, SIM-security implies that

there exists a simulator that, given only the length of  $m$  and the function outputs  $\{f(m)\}$ , but not  $m$  itself, can simulate the role of the challenger in the real game. However, SIM-security is so strong that it suffers from the following impossible results:

- (1) [BSW11]: SIM-secure FE for P/poly cannot be constructed for unbounded ciphertext queries before a single key query;
- (2) [AGVW13]: SIM-secure FE for P/poly cannot be constructed for unbounded key queries before a single ciphertext query.

These impossible results hold even under the random oracle model [AKW18]. Indeed, there are already some constructions for simulation-based FE schemes, but they only work for either bounded ciphertext queries or bounded key queries (which means that the number of ciphertext/key queries must be pre-determined at the Setup phase) [GVW12, GJKS15, ALMT20]. However, for applications in the real world, we need to know how an FE scheme already proven to be SIM-secure performs when handling unbounded ciphertext and key queries. Therefore, a natural question is that: is there a new security notion between IND-security and SIM-security that overcomes the above drawbacks? Intuitively, the new security notion should satisfy the following properties:

- The new security notion must avoid the counter-intuitive example in Example 1.1;
- There must be a construction of FE for P/poly under the new security notion that supports both unbounded ciphertext and key queries;
- Any SIM-secure FE scheme should satisfy the new security notion, so that we are able to discuss the unbounded ciphertext/key security for existing SIM-secure schemes;
- The new security notion should be stronger than IND-security, so that the properties for IND-security also hold for this new security notion.

Next, we show how to define this new security notion by modifying the existing IND-security definition. We note that, the problem in the counter-example can be handled for IND-security, if we loose the restriction on the adversary, such that  $\mathcal{A}$  is still allowed to make query  $f$  even if  $f$  is a one-way permutation. We start from the distributional indistinguishable security (DI-security), first introduced in [AM18], by letting the input of ciphertext queries be a pair of message distributions  $M_0, M_1$ , instead of a pair of messages  $m_0, m_1$ . For example, let  $M_0$  be the uniform distribution of messages such that the first bit is 0, and  $M_1$  be the uniform distribution of messages such that the first bit is 1. When the adversary submits  $M_0, M_1$  to the challenger, the challenger first randomly chooses a bit  $b$ , and then samples  $m \leftarrow M_b$ .

Now we show that why the counter-example can no longer satisfy the DI-security definition which allows probabilistic ciphertexts. We only need to construct an adversary  $\mathcal{A}$  which queries the challenger with a pair of message distributions, instead of a pair of messages, such that  $\mathcal{A}$  can break the scheme in Example 1.1.

- Let  $b$  be a hardcore predicate of  $f$ , we let  $\mathcal{A}$  submit two distributions:  $M_0$  is uniform on all strings with  $b(m) = 0$ ,  $M_1$  is uniform on all strings with  $b(m) = 1$ . (More details can be found in Sect. 5.)
- Now  $\mathcal{A}$  can make queries on  $f$  for the one-way permutation  $f$ , since  $f(M_0)$  and  $f(M_1)$  are computationally indistinguishable by the property of hard core predicate.
- $\mathcal{A}$  can calculate  $\text{PKE.Dec}(sk_f, ct_{m_b})$  and check its first bit to successfully recover  $b$ .

*On computational indistinguishability and queries with trapdoors.* However, DI-security is not enough, mainly because the usage of computational indistinguishability in its security definition. We point out that it is not easy to include computational indistinguishability *inside* a security game. Below we show the difficulties we discovered while attempting to define a new security notion through probabilistic queries, and that how we solved them. We first give an example, where distributional indistinguishability fails to handle.

*Example 1.2.* Let PKE be a public key encryption scheme. We explicitly write the randomness used in the encryption algorithm:  $\text{PKE.Enc}(pk, m; r)$ , let  $\mathcal{R}$  be the space of random seeds where  $r \leftarrow \mathcal{R}$ . We define function class  $\mathcal{F}$  as follows:

$$f_{pk}(m, r) \in \mathcal{F} \Leftrightarrow \exists(pk, sk) \leftarrow \text{PKE.KeyGen}, f_{pk}(m, r) = \text{PKE.Enc}(pk, m; r).$$

Let FE be a functional encryption scheme for  $\mathcal{F}$ , and we consider the security notion which allows message distributions instead of messages.

We construct an adversary  $\mathcal{A}$  which makes following queries:

- $\mathcal{A}$  runs  $\text{PKE.KeyGen}$  to get  $(pk, sk)$ .
- Then,  $\mathcal{A}$  submits  $f_{pk}$  as a key query.
- $\mathcal{A}$  chooses random  $m_0, m_1$ , and submits  $M_0, M_1$  which are uniform distributions on  $\{m_0\} \times \mathcal{R}$  and  $\{m_1\} \times \mathcal{R}$ .

Now we have that  $f_{pk}(M_0) \leftarrow \text{PKE.Enc}(pk, m_0)$  and  $f_{pk}(M_1) \leftarrow \text{PKE.Enc}(pk, m_1)$ , hence the two distributions:  $f(M_0)$  and  $f(M_1)$  are computationally indistinguishable according to the IND-CPA security of PKE. However, the adversary  $\mathcal{A}$  can easily distinguish between a ciphertext in  $f(M_0)$  and  $f(M_1)$  since it holds the secret key  $sk$ .

Although in [AM18], the authors constructed DI-secure FE for all polynomial-sized circuits, we show here that DI-security cannot be satisfied for a function family with trapdoors<sup>1</sup>, which makes a contradictory. The main reason for this problem is that the notion of computational indistinguishability was

<sup>1</sup> We also note that a trapdoor may not only be hidden in the function, but also in the messages. We slightly modify the function in Example 1.2, and let  $f$  be defined as:  $f(pk, m, r) = \text{PKE.Enc}(pk, m; r)$ , and  $M_b = \{pk\} \times \{m_b\} \times \mathcal{R}$ , so  $f(M_0)$  and  $f(M_1)$  are distributions with trapdoor, and the trapdoor is hidden in the message distribution, not the function.

not well-defined as in [AM18]: it must be made clear for which party it is to distinguish between the two distributions, and how much information it has. In Example 1.2, since an adversary may cheat, we cannot let  $\mathcal{A}$  be the distinguisher. However,  $\mathcal{A}$  is the only one who has the secret key  $sk$ , and for any other party,  $f(M_0)$  and  $f(M_1)$  are indistinguishable, which meets the same difficulties.

This is why we must extend computational indistinguishability into a stronger notion for such a security notion of FE to be well-defined. We informally state what it means by saying that two distributions are strictly computationally indistinguishable even considering trapdoors.

**Definition 1.1.** (informal) Let  $\mathcal{D}$  be a p.p.t. algorithm that outputs a pair of distributions  $D_0, D_1$ , we say that distributions from  $\mathcal{D}$  are strictly computationally indistinguishable, if there is no auxiliary string  $aux$  corresponding with  $D_0, D_1$  such that  $(D_0, aux)$  and  $(D_1, aux)$  are computational distinguishable.

Note that the auxiliary string  $aux$  can be viewed as the trapdoor in distributions  $D_0, D_1$ .

Now, we revisit Example 1.2. We consider  $\mathcal{A}$  as the algorithm which outputs  $f_{pk}(M_0)$  and  $f_{pk}(M_1)$  as a pair of distributions, and we let  $sk$  be the auxiliary string  $aux$ . So we can easily construct  $\mathcal{B}$  that distinguish between  $sk, f_{pk}(M_0)$  and  $sk, f_{pk}(M_1)$ , thus  $f_{pk}(M_0)$  and  $f_{pk}(M_1)$  cannot satisfy the condition of strict computational indistinguishability.

This additional auxiliary string has no affect on function families without trapdoors. Just consider PKE, for  $(pk, sk) \leftarrow \text{PKE.KeyGen}$ ,  $sk$  can be the auxiliary string if  $pk$  is fixed, but if we choose random  $pk$ , then there is no such  $aux$  as long as PKE has semantic security (we note that  $aux$  is not a variable, hence cannot be  $sk$ ). Otherwise,  $aux$  becomes a “master trapdoor” which is unrelated to the randomness used in PKE.KeyGen. Let  $\mathcal{A}$  be an adversary which distinguishes between  $(D_0, aux)$  and  $(D_1, aux)$ , then  $\mathcal{A}^{aux}(\cdot) = \mathcal{A}(\cdot, aux)$  (with  $aux$  hardwired in the adversary) can break the semantic security of PKE. We shall give a formal explanation for this case in Sect. 6.

*The need for probabilistic function queries.* It seems that everything is right with a new definition for computational indistinguishability. However, since we consider trapdoor functions, we extend Example 1.2 to construct another example just like Example 1.1:

*Example 1.3.* Let  $\mathcal{F}$  be defined as in Example 1.2, and PKE' be a semantic secure public key encryption scheme, then the following FE construction for  $\mathcal{F}$  is IND-secure even if we consider probabilistic ciphertext queries:

- Setup( $1^\lambda$ ): Let  $(\text{PKE}'.pk^*, \text{PKE}'.sk^*) \leftarrow \text{PKE}'.\text{Setup}(1^\lambda)$ , and returns  $\text{PK} = \text{PKE}'.pk^*$ ,  $\text{MSK} = \text{PKE}'.sk^*$ .
- Enc( $\text{PK}, (m, r)$ ): Return  $\text{PKE}'.\text{Enc}(\text{PK}, m \| r)$ .
- KeyGen( $\text{MSK}, f$ ): Return  $(\text{MSK}, f)$ .
- Dec( $sk_f, ct_m$ ): Parse  $sk_f = (\text{MSK}, f)$ , decrypt  $m \| r = \text{PKE}'.\text{Dec}(\text{MSK}, ct_m)$ , and return  $f(m, r)$ .

However, each  $sk_f$  totally leaks  $m$ , while  $f(m, r)$  does not leak  $m$  (since  $f$  is the encryption of a semantic secure PKE).

The counter-example above holds, since if we allow the adversary to make even a single query, it can first use  $\text{PKE.Setup}$  to generate a pair  $pk, sk$ , and then query the function key for  $f_{pk} = \text{PKE.Enc}(pk, \cdot; \cdot) \in \mathcal{F}$ , hence having the ability to trivially distinguish between  $f_{pk}(M_0)$  and  $f_{pk}(M_1)$ . (To match Definition 1.1 above, we can trivially construct a distinguisher  $\mathcal{B}$  with  $sk$  as the auxiliary string.) Since the adversary cannot make any queries, the same problem in Example 1.1 also occurs.

In order to avoid such counter-examples, we must allow probabilistic queries not only in the ciphertext query, but also in key queries. Each time the adversary makes a probabilistic key query  $F$ , the challenger first samples  $f \leftarrow F$ , then returns both  $f$  and  $sk_f$  to the adversary. We construct an adversary  $\mathcal{A}$  which makes following queries (including probabilistic key queries):

- We let  $\mathcal{A}$  submit two distributions:  $M_0$  is uniform on all strings which first bit is 0,  $M_1$  is uniform on all strings which first bit is 1.
- $\mathcal{A}$  makes a single key query by submitting a distribution  $F$  which is uniform on  $\mathcal{F}$ , and gets  $f_{pk} \in \mathcal{F}$ .
- $\mathcal{A}$  can calculate  $\text{PKE'.Dec}(\text{MSK}, ct_{m_b})$  and check its first bit to successfully recover  $b$ .

Since  $f_{pk}$  is randomly chosen by the challenger, the adversary  $\mathcal{A}$  cannot get the corresponding  $sk$ . Here, instead of  $f(M_0)$  and  $f(M_1)$ , we only require that the distributions  $F, F(M_0)$  and  $F, F(M_1)$  be strictly computationally indistinguishable (sampling from  $F, F(M_b)$  means sampling  $f \leftarrow F, m \leftarrow M_b$  and returning  $f, f(m)$ .) By our definition, the auxiliary string  $aux$  is only related to the distribution  $F, F(M_b)$  but independent from how the challenger chooses  $f_{pk} \leftarrow F$  (thus independent with either  $pk$  or  $sk$ ).

Now we finished the discussion of rationality for probabilistic queries. We can see that such a security notion can be well-defined, and also avoids the counter-intuitive examples in Example 1.1, 1.2 and 1.3. We call the new security notion *indistinguishability-based security against probabilistic queries* (pIND-security), and show that it is weaker than SIM-security but stronger than IND-security.

*Construction of pIND-secure FE for P/poly.* In this paper, we also give a construction of pIND-secure FE for P/poly, which allows unbounded number of both ciphertext and key queries. Concretely, we show that a fully pIND-secure FE scheme for P/poly can be constructed from a selective IND-secure FE scheme, while the latter can be constructed from both indistinguishability obfuscation [GGH+13] and well-founded assumptions [JLS21, GP21, WW21]. We note that, although the existence of  $i\mathcal{O}$  is a strong assumption, unbounded IND-secure FE for P/poly is more than sufficient in constructing  $i\mathcal{O}$  [AJ15]. So the FE scheme we construct has stronger security without stronger assumptions.

## 1.2 Related Works

*FE for randomized functionalities.* Functional encryption for randomized functionalities (rFE) was introduced in [GJKS15]. The authors gave both SIM-based



and IND-based security notions for rFE. We note that, since the authors also used computational indistinguishability to define IND-based security for rFE, the same problems occur as we pointed out in Example 1.2, so that the IND-security of rFE given by [GJKS15] cannot handle trapdoors or public-key encryption. By moving our definition (and construction, using the generic transformation of [AW17]) into the randomized case, these problems can be solved to get a well-defined pIND-based security for rFE.

*Distributional Indistinguishability for FE.* In [AM18], the authors gave the definition of distributional indistinguishability (DI) for FE, which is previously discussed on garbled circuit and randomized encodings [GHRW14,LPST16], and also gave a construction for DI-secure FE from standard IND-secure FE. Our security definition shares some similarities with theirs, such as allowing the adversary to submit two message distributions, rather than two messages, in the ciphertext query. However, since the DI definition does not allow probabilistic key queries, it still suffers from Example 1.2 and 1.3 which we pointed out above (see also the discussion in Sect. 6). Moreover, we also give a pIND-secure FE construction for P/poly with adaptive security, while the construction in [AM18] only satisfies selective security.

*Function-private public key FE.* Probabilistic key queries are also considered in the function-privacy of public key FE [BRS13,PMR19,BCJ+19] as in our work. However, we do not consider function-privacy: In our definition, the function chosen by the challenger is always known to the adversary. It is interesting that whether we can extend our security definition to handle function-privacy.

*Other security definitions for FE.* There are some other security definitions in early works of functional encryption. In [BO13], the authors gave some new security definitions compared with IND-security, but without a general construction. In [BF13], the authors considered the cases where a trapdoor is hidden in the function family  $\mathcal{F}$  supported by FE (instead of function-key queries, which we consider in this paper), and made new security definitions that are even stronger than SIM-security. However, in this paper, we mainly focus on the case where the function family  $\mathcal{F}$  is P/poly, so we will not consider this problem.

## 2 Preliminaries

*Notations .*  $x \leftarrow \chi$  for a distribution  $\chi$  means that  $x$  is sampled from  $\chi$ .  $x \leftarrow X$  for a set  $X$  means that  $x$  is uniformly random chosen from  $X$ .  $x \leftarrow \mathcal{X}$  for a p.p.t. algorithm  $\mathcal{X}$  means that  $x$  is a random output of  $\mathcal{X}$ , where the abbreviation p.p.t. stands for probabilistic polynomial time. We say that  $\epsilon$  is negligible in  $\lambda$ , if  $\epsilon < 1/\Omega(\lambda^c)$  for any  $c > 0$  with sufficiently large  $\lambda$ .  $[n]$  for  $n \in \mathbb{Z}_+$  is the set  $\{1, \dots, n\}$ .

### 2.1 Functional Encryption and Security Definitions

**Definition 2.1.** *A functional encryption scheme FE for a function family  $\mathcal{F}$  consists of the following four algorithms (let  $\mathcal{M}$  be the message space):*

- $\text{Setup}(1^\lambda)$ : output a pair  $(\text{PK}, \text{MSK})$ .
- $\text{KeyGen}(\text{MSK}, f)$ : for  $f \in \mathcal{F}$ , output a function key  $\text{SK}_f$ .
- $\text{Enc}(\text{PK}, m)$ : for  $m \in \mathcal{M}$ , output a ciphertext  $\text{CT}_m$ .
- $\text{Dec}(\text{SK}_f, \text{CT}_m)$ : output the function value  $f(m)$ .

FE is correct if for any  $(\text{PK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ ,  $\text{SK}_f \leftarrow \text{KeyGen}(\text{MSK}, f)$ ,  $\text{CT}_m \leftarrow \text{Enc}(\text{PK}, m)$ , the probability that  $\text{Dec}(\text{SK}_f, \text{CT}_m) \neq f(m)$  is negligible.

Now we give the definition for both IND-security and SIM-security of functional encryption.

**Definition 2.2.** An 1-CT adaptive IND-CPA-security game for an FE scheme is defined as follows:

- Setup: The challenger runs  $\text{Setup}(1^\lambda)$  and returns  $\text{PK}$  to the adversary.
- Phase 1: The adversary chooses  $f \in \mathcal{F}$  and gives it to the challenger. The challenger generates  $sk_f \leftarrow \text{KeyGen}(\text{MSK}, f)$  and returns  $sk_f$  to the adversary. This can be repeated adaptively for any polynomial times.
- Challenge: The adversary chooses two messages of identical length  $m_0, m_1$  and gives it to the challenger. The challenger randomly chooses  $b \leftarrow \{0, 1\}$ , generates  $ct \leftarrow \text{Enc}(\text{PK}, m_b)$  and returns  $ct$  to the adversary.
- Phase 2: Same as Phase 1.
- Output: The adversary outputs a bit  $b'$ , and the winning advantage for the adversary is defined by  $\text{Adv}^{\text{IND}}(\mathcal{A}) = |\Pr(b' = b) - 1/2|$ .

An adversary  $\mathcal{A}$  is said to be admissible, if for any query  $f$  in Phase 1 or Phase 2,  $f(m_0) = f(m_1)$ . FE is said to be ad-IND-secure if for any p.p.t. admissible adversary  $\mathcal{A}$ ,  $\text{Adv}^{\text{IND}}(\mathcal{A})$  is negligible.

For the selective IND-security (sel-IND-security), we require that  $\mathcal{A}$  submits  $m_0, m_1$  to the challenger at the beginning of the game.

For the many-CT version of the game, we let the adversary submits any polynomial number of pairs of messages in the challenge phase, say  $(m_0^1, m_1^1), \dots, (m_0^q, m_1^q)$ , such that  $f(m_0^i) = f(m_1^i)$  for any query  $f$  and  $i \in [q]$ . In the challenge phase, the challenger samples  $b \leftarrow \{0, 1\}$  and returns  $(\text{Enc}(\text{PK}, m_b^i))_{i \in [q]}$ .

It is not hard to show that 1-CT IND-security implies many-CT IND-security through hybrid arguments. In [ABSV15], the authors showed that any sel-IND secure FE scheme which is sufficiently expressive can be turned into an ad-IND secure FE scheme. Even if the FE scheme is not expressive enough, we can still use the standard complexity leverage method [BB04] to prove the ad-IND-security, if we assume the sub-exponential hardness of the underlying hardness assumptions.

Next, we give the simulation-based security definition.

**Definition 2.3.** Let FE be a functional encryption scheme for a function family  $\mathcal{F}$ . Consider a p.p.t. adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  and a stateful p.p.t. simulator  $\text{Sim}$ . Let  $U_m(\cdot)$  denote a universal oracle, such that  $U_m(f) = f(m)$ . Consider the following two experiments:

$\text{Exp}_{\text{FE}, \mathcal{A}}^{\text{real}}(1^\lambda)$	$\text{Exp}_{\text{FE}, \mathcal{A}}^{\text{ideal}}(1^\lambda)$
1. $(\text{PK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\lambda)$ ;	1. $\text{PK} \leftarrow \text{Sim}(1^\lambda)$ ;
2. $(m, st) \leftarrow \mathcal{A}_1^{\text{FE.KeyGen}(\text{MSK}, \cdot)}(\text{PK})$ ;	2. $(m, st) \leftarrow \mathcal{A}_1^{\text{Sim}(\cdot)}(\text{PK})$ ;
3. $\text{CT} \leftarrow \text{FE.Enc}(\text{PK}, m)$ ;	3. $\text{CT} \leftarrow \text{Sim}^{U_m(\cdot)}(1^\lambda, 1^{ m })$ ;
4. $\alpha \leftarrow \mathcal{A}_2^{\text{FE.KeyGen}(\text{MSK}, \cdot)}(\text{PK}, \text{CT}, st)$ ;	4. $\alpha \leftarrow \mathcal{A}_2^{\text{Sim}^{U_m(\cdot)}(\cdot)}(\text{PK}, \text{CT}, st)$ ;
5. Output $m, \alpha$	5. Output $m, \alpha$

We call a stateful simulator algorithm  $\text{Sim}$  admissible if, on each input  $f$ ,  $\text{Sim}$  makes just a single query to its oracle  $U_m(\cdot)$  on  $f$  itself. The functional encryption scheme  $\text{FE}$  is then said to be adaptive simulation-based secure (ad-SIM-secure) if there is an admissible stateful p.p.t. simulator  $\text{Sim}$  such that for every p.p.t. adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ , the two experiments are computationally indistinguishable.

For the selective SIM-security (sel-SIM-security), we require that  $\mathcal{A}$  submits  $m$  to the challenger at the beginning of the game.

### 3 Indistinguishability-Based Security Against Probabilistic Queries

#### 3.1 Definition for pIND Security

First, we give a formal definition for the idea of strict computational indistinguishability introduced in Sect. 1. We say that a distribution  $F$  is efficiently samplable, if there exists a p.p.t. algorithm  $\mathcal{F}$  which output follows  $F$ . Moreover, sampling from  $F$  means to run  $\mathcal{F}$  with random seed and fetch its output, so we can use  $\mathcal{F}$  to represent  $F$  if there is no confusion.

**Definition 3.1.** Let  $\mathcal{D}$  be a p.p.t. algorithm that outputs a pair of efficiently samplable distributions  $D_0, D_1$ . We say that distributions from  $\mathcal{D}$  are strictly computationally indistinguishable, if for any p.p.t. algorithm  $\mathcal{S}$  which outputs a pair of efficiently samplable distributions  $S_0, S_1$  and an auxiliary string  $aux$ , either:

- (1) there exists a p.p.t. algorithm  $\mathcal{P}$  which distinguishes between the output of  $\mathcal{D}$  and  $\mathcal{S}$  (without  $aux$ ), which means that  $\Pr(\mathcal{P}(D_0, D_1) = 1 | (D_0, D_1) \leftarrow \mathcal{D}) - \Pr(\mathcal{P}(S_0, S_1) = 1 | (S_0, S_1, aux) \leftarrow \mathcal{S})$  is non-negligible;
- or
- (2) there is no p.p.t. algorithm  $\mathcal{B}$  which distinguishes between  $aux, S_0$  and  $aux, S_1$ , which means that  $\Pr(\mathcal{B}(aux, s_0) = 1 | s_0 \leftarrow S_0) - \Pr(\mathcal{B}(aux, s_1) = 1 | s_1 \leftarrow S_1)$  must be negligible.

Without loss of generality, we let the auxiliary string  $aux$  contains the two distributions  $S_0, S_1$  (in the form of sampling algorithms), so the distinguisher  $\mathcal{B}$  knows exactly the two distributions.

Since there must be no restriction on how  $\mathcal{D}$  works, we cannot suppose that  $aux$  is output by  $\mathcal{D}$ , hence we introduce another algorithm  $\mathcal{S}$  which outputs both the pair of distributions and the auxiliary string  $aux$ . In most cases, we can simply suppose that  $\mathcal{S}$  acts similar as  $\mathcal{D}$ . However, since no p.p.t. algorithm can determine whether two distributions are equal or even statistical indistinguishable, in order to get a formal definition, we simply let the outputs of  $\mathcal{D}$  and  $\mathcal{S}$  be computationally indistinguishable.

Next, we use the idea of strict computational indistinguishability to define our new definition for functional encryption.

**Definition 3.2.** *Given message space  $\mathcal{M}$  and function space  $\mathcal{F}$ , an 1-CT adaptive pIND-CPA-security game for an FE scheme is defined as the following:*

- Setup: The challenger runs  $\text{Setup}(1^\lambda)$  and returns PK to the adversary.
- Phase 1: The adversary chooses an efficiently samplable distribution  $F$  on the function space  $\mathcal{F}$ , and gives the sampling algorithm to the challenger. The challenger samples  $f \leftarrow F$ , generates  $sk_f = \text{KeyGen}(\text{MSK}, f)$  and returns  $f, sk_f$  to the adversary. This can be repeated adaptively for any polynomial times.
- Challenge: The adversary chooses two efficiently samplable distributions  $M_0, M_1$  on the message space  $\mathcal{M}$  which contain messages of same length, and gives the sampling algorithms to the challenger. The challenger randomly chooses  $b \leftarrow \{0, 1\}$ ,  $m \leftarrow M_b$ , generates  $ct_m \leftarrow \text{Enc}(\text{PK}, m)$  and returns  $ct_m$  to the adversary.
- Phase 2: Same as Phase 1.
- Output: The adversary outputs  $b'$ , and the winning advantage for the adversary is defined by  $\text{Adv}^{\text{pIND}}(\mathcal{A}) = |\Pr(b' = b) - 1/2|$ .

An adversary  $\mathcal{A}$  is said to be admissible, if the two distributions  $(F_i, F_i(M_0))_{i \in [Q]}$  and  $(F_i, F_i(M_1))_{i \in [Q]}$  are strictly computationally indistinguishable,  $Q$  is the number of KeyGen queries. FE is said to be ad-pIND-secure if for any p.p.t. admissible adversary  $\mathcal{A}$ ,  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is negligible.

For the selective pIND-security (sel-pIND-security), we require that  $\mathcal{A}$  submits  $M_0, M_1$  to the challenger at the beginning of the game.

For the many-CT version of the game, we let the adversary submits any polynomial number of pairs of messages in the challenge phase, say  $(M_0^1, M_1^1), \dots, (M_0^q, M_1^q)$ , and the admissability is changed to:  $(F_i, F_i(M_0^1), \dots, F_i(M_0^q))_{i \in [Q]}$  and  $(F_i, F_i(M_1^1), \dots, F_i(M_1^q))_{i \in [Q]}$  are strictly computationally indistinguishable. In the challenge phase, the challenger samples  $b \leftarrow \{0, 1\}$  and returns  $(\text{Enc}(\text{PK}, m_b^i))_{i \in [q]}$ .

We note that when sampling from the distribution  $(F_i, F_i(M_b))_{i \in [Q]}$ , we only sample once from each  $F_i$  and  $M_b$ , so the elements from the distribution are in fact dependent with each other.

Now we present a lemma by applying the contrapositive of strict computational indistinguishability onto pIND-security definition. This lemma is useful in the following proofs.

**Lemma 3.1.** *For an adversary  $\mathcal{A}$  in the pIND-CPA-security game, we define the trace of  $\mathcal{A}$  as:*

$$tr_{\mathcal{A}} = (M_0, M_1, (F_i, f_i, f_i(m))_{i \in [Q]}).$$

*Then FE is pIND-secure, if and only if for every p.p.t.  $\mathcal{A}$  such that  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is non-negligible (not necessarily admissible), there exists a p.p.t. sampling algorithm  $\mathcal{T}$  which outputs the distribution:*

$$(aux, \bar{b} \leftarrow \{0, 1\}, \bar{m} \leftarrow \bar{M}_{\bar{b}}, \bar{tr} = (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}))_{i \in [Q]})),$$

*and a p.p.t. algorithm  $\mathcal{B}$  where:*

- (1) *For any p.p.t. algorithm  $\mathcal{P}$ ,  $\Pr(\mathcal{P}(tr_{\mathcal{A}}) = 1) - \Pr(\mathcal{P}(\bar{tr}) = 1)$  is negligible;*
- (2)  *$aux$  is independent with the following conditional distributions:  $\bar{m} | \bar{M}_0, \bar{M}_1; \bar{f}_1 | \bar{F}_1; \dots; \bar{f}_Q | \bar{F}_Q$  (which can be considered as the randomness used in the choice of  $\bar{m}, \bar{f}_1, \dots, \bar{f}_Q$ );*
- (3)  *$\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is non-negligible.*

*Proof.* If FE is pIND-secure, then  $\mathcal{A}$  with non-negligible advantage must be non-admissible, which means that  $(F_i, F_i(M_0))_{i \in [Q]}$  and  $(F_i, F_i(M_1))_{i \in [Q]}$  are not strictly computationally indistinguishable.

By the definition of strict computational indistinguishability, there is a sampling algorithm  $\mathcal{S}$  which outputs  $(\bar{F}_i, \bar{F}_i(\bar{M}_0))_{i \in [Q]}, (\bar{F}_i, \bar{F}_i(\bar{M}_1))_{i \in [Q]}, aux$ , such that:

- (1) The output of  $\mathcal{S}$  except  $aux$  are computationally indistinguishable with  $(F_i, F_i(M_0))_{i \in [Q]}, (F_i, F_i(M_1))_{i \in [Q]}$ ;
- (2) There exists  $\mathcal{B}$  which distinguishes between  $aux, (\bar{F}_i, \bar{F}_i(\bar{M}_0))_{i \in [Q]}$  and  $aux, (\bar{F}_i, \bar{F}_i(\bar{M}_1))_{i \in [Q]}$ .

Let  $\mathcal{T}$  do the following: first sample  $S_0, S_1, aux$  from  $\mathcal{S}$ , then sample  $\bar{b} \leftarrow \{0, 1\}, \bar{f}_i \leftarrow \bar{F}_i, \bar{m} \leftarrow \bar{M}_{\bar{b}}$ , and return  $(aux, \bar{b}, \bar{m}, \bar{tr} = (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}))_{i \in [Q]}))$ . Since in the pIND-CPA-security game, the challenger samples from  $F_i$  and  $M_b$  honestly, we can see that  $\bar{tr}$  is computationally indistinguishable with  $tr_{\mathcal{A}}$ , and  $aux$  is independent with the choice of  $\bar{f}_i$  and  $\bar{m}$ , which means that  $aux$  is independent with  $\bar{m} | \bar{M}_0, \bar{M}_1; \bar{f}_1 | \bar{F}_1; \dots; \bar{f}_Q | \bar{F}_Q$ , hence satisfies all three conditions.

Now, suppose that there exists  $\mathcal{T}, \mathcal{B}$  satisfies all three conditions. Let  $\mathcal{S}$  runs  $\mathcal{T}$  and outputs  $aux$  and the two distributions  $(\bar{F}_i, \bar{F}_i(\bar{M}_0))_{i \in [Q]}, (\bar{F}_i, \bar{F}_i(\bar{M}_1))_{i \in [Q]}$ , which are computationally indistinguishable with  $(F_i, F_i(M_0))_{i \in [Q]}, (F_i, F_i(M_1))_{i \in [Q]}$ .

Then, we sample random  $\bar{b} \leftarrow \{0, 1\}, \bar{m} \leftarrow \bar{M}_{\bar{b}}, \bar{f}_i \leftarrow \bar{F}_i, i \in [Q]$ , and let  $(aux, (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}))_{i \in [Q]}))$  be the input of  $\mathcal{B}$ , then  $\mathcal{B}$  distinguishes the two distributions  $(\bar{F}_i, \bar{F}_i(\bar{M}_0))_{i \in [Q]}, (\bar{F}_i, \bar{F}_i(\bar{M}_1))_{i \in [Q]}$ . By the definition of strict computational indistinguishability,  $(F_i, F_i(M_0))_{i \in [Q]}$  and  $(F_i, F_i(M_1))_{i \in [Q]}$  cannot be strictly computationally indistinguishable, which means that any adversary  $\mathcal{A}$  with non-negligible advantage cannot be admissible.  $\square$

For the many-CT version of the game, we define the trace  $tr_{\mathcal{A}}$  as:

$$((M_0^i, M_1^i)_{i \in [q]}, (F_k, f_k, f_k(m_1), \dots, f_k(m_q))_{k \in [Q]}).$$

It is not hard to show that the result is the same as the 1-CT case.

In a general case, it seems to be hard to determine whether two distributions are strictly computationally indistinguishable, especially with the auxiliary string. But if the function class is a cryptographic primitive such as hash family or public key encryption, we can use its security definition to prove the indistinguishability. We give more details in Sect. 5 and Sect. 6.

### 3.2 Relationship Between Different Security Definitions

In this section, we show that pIND-security satisfies the four properties we discussed in Sect. 1.1, which means that pIND-security can be used to avoid the drawbacks for both SIM-security and IND-security.

**Theorem 3.1.** *If FE is SIM-secure, then FE is pIND-secure.*

*Proof.* Let  $\mathcal{A}$  be any pIND adversary, we can construct a SIM adversary  $\mathcal{E}^{\mathcal{A}}$  as follows (for the real experiment):

- When  $\mathcal{A}$  outputs a key query  $F$ ,  $\mathcal{E}$  chooses  $f \leftarrow F$  and gives  $f$  to the challenger  $\mathcal{C}$ . When the challenger returns  $sk_f$ ,  $\mathcal{E}$  returns  $f, sk_f$  to  $\mathcal{A}$ .
- When  $\mathcal{A}$  outputs the ciphertext query  $M_0, M_1$ ,  $\mathcal{E}$  first chooses  $b \leftarrow \{0, 1\}$  and gives  $m \leftarrow M_b$  to the challenger  $\mathcal{C}$ .
- When  $\mathcal{C}$  returns a ciphertext  $CT$ ,  $CT$  is returned to  $\mathcal{A}$  directly.
- When  $\mathcal{A}$  outputs the guess  $b'$ ,  $\mathcal{E}$  outputs  $b'$  along with the trace:  $tr_{\mathcal{A}} = (M_0, M_1, (F_1, f_1, f_1(m)), \dots, (F_Q, f_Q, f_Q(m)))$ .

Since  $b'$  is the same as the output of  $\mathcal{A}$  in the sel-pIND game,  $\Pr(b' = b) - 1/2$  is non-negligible iff  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is non-negligible.

Now consider the ideal experiment with simulator  $\mathcal{S}$ . Differ from the real experiment, we let the random bit and sampled message be  $\tilde{b}, \tilde{m}$ , output be  $\tilde{b}'$ , the trace by  $\tilde{tr}_{\mathcal{A}}$ , and  $\tilde{tr}_{\mathcal{A}}$  is computationally indistinguishable with  $tr_{\mathcal{A}}$  by the SIM-based security of the FE scheme. So  $\Pr(\tilde{b}' = \tilde{b}) - 1/2$  is non-negligible iff  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is non-negligible.

Using Lemma 3.1, we only need to construct the algorithm  $\mathcal{B}$  and a sampling algorithm  $\mathcal{T}$  which samples  $(aux, \tilde{b}, \tilde{m}, \tilde{tr})$ .

Let  $\mathcal{T}$  run the ideal experiment with adversary  $\mathcal{E}^{\mathcal{A}}$  and simulator  $\mathcal{S}^{U_{\tilde{m}}(\cdot)}$ . When  $\mathcal{S}$  queries  $U_{\tilde{m}}(f)$ , it directly returns  $f(\tilde{m})$  to  $\mathcal{S}$  (since  $\tilde{m}$  is chosen by  $\mathcal{E}^{\mathcal{A}}$ ), and let  $\tilde{b} = \tilde{b}, \tilde{m} = \tilde{m}, \tilde{tr} = \tilde{tr}_{\mathcal{A}}$ . Finally, let  $aux = (r_{\mathcal{A}}, r_{\mathcal{S}})$ , where  $r_{\mathcal{A}}, r_{\mathcal{S}}$  are the randomness used in  $\mathcal{A}, \mathcal{S}$ .

$\mathcal{B}(aux = (r_{\mathcal{A}}, r_{\mathcal{S}}), \tilde{tr} = (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \bar{f}_i, \bar{f}_i(\tilde{m}))_{i \in [Q]}))$  is constructed from  $\mathcal{A}, \mathcal{S}$  with  $r_{\mathcal{A}}, r_{\mathcal{S}}$  as their randomness:

- $\mathcal{B}$  first runs  $\mathcal{A}$  with randomness  $r_{\mathcal{A}}$ . When  $\mathcal{A}$  outputs the ciphertext query  $\tilde{M}_0, \tilde{M}_1$ , first check  $\tilde{M}_0 = \bar{M}_0, \tilde{M}_1 = \bar{M}_1$ , otherwise abort.  $\mathcal{S}$  is run with randomness  $r_{\mathcal{S}}$ .

- When  $\mathcal{A}$  outputs the  $i$ -th key query  $\tilde{F}_i$ , first check  $\tilde{F}_i = \bar{F}_i$ , otherwise abort. Send  $\tilde{f}_i$  to  $\mathcal{S}$ , and when  $\mathcal{S}$  queries  $U_{\tilde{m}}$ , return  $\tilde{f}_i(\tilde{m})$  to  $\mathcal{S}$ . Return  $\tilde{f}_i$  and  $sk_{\tilde{f}_i}$  generated by  $\mathcal{S}$  to  $\mathcal{A}$ .
- When  $\mathcal{S}$  returns a ciphertext  $CT$ ,  $CT$  is returned to  $\mathcal{A}$  directly.
- When  $\mathcal{A}$  outputs the guess  $\tilde{b}'$ , return  $\tilde{b}' = \tilde{b}$ .

It is easy to see that if  $\mathcal{B}$  never aborts, the output distribution is the same as  $\mathcal{E}^{\mathcal{A}}$  in the ideal game, which means that  $\Pr(\mathcal{B}(aux, \bar{tr}) = b) - 1/2$  is non-negligible iff  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is non-negligible, hence FE satisfies pIND-security. The non-abortness directly follows from the fact that the queries from  $\mathcal{A}$  in both  $\mathcal{B}$  and  $\mathcal{T}$  are uniquely determined by the same randomness used by  $\mathcal{A}, \mathcal{E}, \mathcal{S}$ , so that  $\bar{M}_0, \bar{M}_1, \bar{F}_1, \dots, \bar{F}_Q$  in  $\mathcal{B}$  are exactly the same as  $\bar{M}_0, \bar{M}_1, \bar{F}_1, \dots, \bar{F}_Q$  contained in  $\bar{tr}$  generated from  $\mathcal{T}$ . Thus we finish the proof.  $\square$

**Theorem 3.2.** *If FE is pIND-secure, then FE is IND-secure.*

*Proof.* For any admissible IND adversary  $\mathcal{A}$ , we construct a pIND adversary  $\mathcal{A}'$  as follows:

- When  $\mathcal{A}$  submits  $m_0, m_1$ ,  $\mathcal{A}'$  submits  $M_0, M_1$  such that  $M_b(m_b) = 1$ ,  $M_b(m') = 0$  for  $m' \neq m_b$ ,  $b \in \{0, 1\}$ .
- When  $\mathcal{A}$  submits  $f$ ,  $\mathcal{A}'$  submits  $F$  such that  $F(f) = 1$ ,  $F(f') = 0$  for  $f' \neq f$ ,  $f(m_0) = f(m_1)$ .
- When  $\mathcal{A}$  outputs a bit  $b'$ ,  $\mathcal{A}'$  also outputs  $b'$ .

If  $\text{Adv}^{\text{pIND}}(\mathcal{A}')$  is non-negligible, then there exists  $\mathcal{B}$ ,  $aux$  and  $\bar{tr} = (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \tilde{f}_i, \tilde{f}_i(\tilde{m}))_{i \in [Q]})$ , such that  $\Pr(\mathcal{B}(aux, \bar{tr}) = \tilde{b}) - 1/2$  is non-negligible. Also,  $\bar{tr}$  is indistinguishable from  $tr_{\mathcal{A}}$ , which means that sampling from  $\bar{M}_0, \bar{M}_1$  and  $\bar{F}_i$  always outputs fixed values  $\tilde{m}_0, \tilde{m}_1, \tilde{f}_i$ , where  $\tilde{f}_i(\tilde{m}_0) = \tilde{f}_i(\tilde{m}_1)$  for  $i \in [Q]$  (otherwise  $tr_{\mathcal{A}}$  and  $\bar{tr}$  can easily be distinguished). So  $\bar{tr}$  is independent from  $\tilde{b}$ , also  $aux$  is independent from  $\tilde{b}$  by Lemma 3.1. Thus  $\Pr(\mathcal{B}(aux, \bar{tr}) = \tilde{b}) - 1/2 = 0$ , which makes a contradiction.

So for every  $\mathcal{A}'$  defined above,  $\text{Adv}^{\text{pIND}}(\mathcal{A}')$  is negligible, which means that  $\text{Adv}^{\text{IND}}(\mathcal{A})$  is negligible.  $\square$

Now we show that 1-CT pIND-security implies many-CT pIND-security, so that our new definition can really bypass the impossible result in [BSW11].

**Theorem 3.3.** *If FE is 1-CT pIND-secure, then FE is many-CT pIND-secure.*

*Proof.* We define a sequence of games:

$G_i$ : the first  $i$  ciphertext queries always choose  $m_i \leftarrow M_0$  despite whether  $b$  is. Suppose that  $\mathcal{A}$  makes a total of  $q$  ciphertext queries, then  $G_0$  is the original game, and the advantage for  $\mathcal{A}$  in  $G_q$  is always 0.

If the advantage for  $\mathcal{A}$  in  $G_0$  is non-negligible, then there exists  $i \in [q]$  such that the advantage of  $\mathcal{A}$  to distinguish between  $G_{i-1}$  and  $G_i$  is non-negligible. Then we construct an 1-CT pIND adversary  $\mathcal{A}_i$  as follows:

- For  $(M_0^j, M_1^j)_{j \in [q]}$ , we define  $M_0(x), M_1(x)$  be two sampling algorithms with a single input  $x \in [q]$ , which sample from  $M_0^x$  and  $M_1^x$ . Thus  $M_0(x), M_1(x)$  contains all information about  $(M_0^j, M_1^j)_{j \in [q]}$ .
- When  $\mathcal{A}$  submits  $(M_0^j, M_1^j)_{j \in [q]}$ , submit  $M_0(i), M_1(i)$  to the challenger and get the ciphertext  $CT$ ; sample  $m_j \leftarrow M_0^j$  for  $j < i$ ,  $m_j \leftarrow M_1^j$  for  $j > i$ , let  $CT_j \leftarrow \text{Enc}(\text{PK}, m_j)$  for  $j \neq i$  and  $CT_i = CT$ , return  $(CT_1, \dots, CT_q)$  to  $\mathcal{A}$ .
- When  $\mathcal{A}$  submits a key query  $F$ , directly pass it to the challenger and return  $(f, sk_f)$  to  $\mathcal{A}$ .
- When  $\mathcal{A}$  outputs  $b'$ , output  $b'$ .

So  $\text{Adv}^{\text{pIND}}(\mathcal{A}_i)$  is non-negligible. By the 1-CT pIND security, there exist  $\mathcal{T}_i$  and  $\mathcal{B}_i$  satisfying Lemma 3.1, let  $(aux, \bar{tr}_i)$  be sampled by  $\mathcal{T}_i$ , we write  $\bar{tr}_i = (\bar{M}_0(i), \bar{M}_1(i), (\bar{F}_k, \bar{f}_k, \bar{f}_k(\bar{m}_i))_{k \in [Q]})$ , and since  $\bar{M}_0(i), \bar{M}_1(i)$  are indistinguishable from  $M_0(i), M_1(i)$ , we write the  $q$  pairs of distributions extracted from  $\bar{M}_0(i), \bar{M}_1(i)$  as  $(\bar{M}_0^j, \bar{M}_1^j)_{j \in [q]}$ .

We first sample  $(\bar{m}_j \leftarrow \bar{M}_b^j)_{j \neq i}$  and calculate  $(\bar{f}_k(\bar{m}_j))_{j \neq i, k \in [Q]}$ . Let  $\mathcal{B}$  proceed the same as  $\mathcal{B}_i$  except that we let the input  $\bar{tr} = ((\bar{M}_0^j, \bar{M}_1^j)_{j \in [q]}, (\bar{F}_k, \bar{f}_k, \bar{f}_k(\bar{m}_1), \dots, \bar{f}_k(\bar{m}_q))_{k \in [Q]})$ . So  $(aux, \bar{b}, (\bar{m}_i)_{i \in [q]}, \bar{tr})$  can be sampled by  $\mathcal{T}_i$  with slight modification,  $aux$  is independent from the choices of  $(\bar{m}_j)_{j \in [q]}$  and  $(\bar{f}_k)_{k \in [Q]}$ , and  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is non-negligible, since the outputs of  $\mathcal{B}$  and  $\mathcal{B}_i$  are the same. Thus we finish the proof.  $\square$

## 4 Fully pIND-secure FE from IND-based FE Schemes

We already show that pIND-secure FE can support unbounded ciphertext. The problem remaining is to show the existence of adaptive pIND-secure FE scheme for P/poly which supports unbounded key, so that our new security definition can avoid the [AGVW13] impossibility result. We show that the [ABSV15] generic transformation, which transforms selective IND-secure FE schemes into adaptive IND-secure ones, can be extended into pIND-security. In fact, we prove a result stronger than expected: we can transform any selective IND-secure FE scheme into an adaptive pIND-secure FE scheme.

*Technical Overview.* In [ABSV15], the authors constructed an adaptive IND-secure FE scheme for any function class  $\mathcal{F}$  (even if  $\mathcal{F} = \text{P/poly}$ ) from an IND-secure private-key FE scheme for  $\mathcal{F}$  with 1-CT query and unbounded key queries, and a “sufficiently expressive” selective IND-secure FE scheme, here private-key FE means that the encryption algorithm uses master secret key instead of master public key.

To prove the existence of IND-secure private-key FE with 1-CT and unbounded key queries, [ABSV15] relies on several results in the literature. First, in [GVW12], the authors constructed a 1-key, unbounded-CT SIM-secure private-key FE scheme for P/poly, which is also a 1-key, unbounded-CT IND-secure private-key FE scheme. In [BS15], the authors gave the generic transformation from private-key FE to function-private private-key FE, here function-private means that the function  $f$  is hidden from the adversary even given



the function key  $sk_f$ . (A private-key FE without function-privacy is also called message-private.) Then, one can swap  $\text{KeyGen}$  and  $\text{Enc}$  in a function-private private-key FE with 1-key and unbounded-CT, to obtain a private-key FE with unbounded-key and 1-CT.

The same method can easily be extended to pIND-security. Similar with [ABSV15], we can construct an adaptive pIND-secure FE scheme from pIND-secure private-key FE scheme and IND-secure (public-key) FE scheme, and by Theorem 3.1 in this paper (extended to private-key settings), we can show the existence of a 1-key, unbounded-CT message-private pIND-secure private-key FE scheme for  $\text{P/poly}$ . What left for us is to transform a pIND-secure message-private private-key FE scheme into a pIND-secure function-private private-key FE scheme.

The idea of this construction is similar to the one in [BS15], but more complicated since we consider probabilistic queries. The [BS15] construction used two symmetric keys  $k, k'$  to hide the two functions  $f_0, f_1$  correspondingly in both the message-private and function-private game. However, in our pIND-secure settings, in message-private game, the adversary learns an exact function  $f$ , while in function-private game, the adversary learns only two distributions  $F_0$  and  $F_1$  (see the formal definition below). So we need three keys  $k, k', k''$  to encrypt  $f, F_0, F_1$  correspondingly, and an additional game to switch between them, while the other parts of the proof is similar to [BS15].

Finally, combining all components together, we can construct an adaptive pIND-secure FE scheme for  $\text{P/poly}$ .

Before further discussions, first we give formal definitions for both message-private and function-private private-key functional encryption with pIND-security.

**Definition 4.1.** *A private-key functional encryption scheme  $\text{skFE}$  for a function family  $\mathcal{F}$  consists of the following four algorithms (let  $\mathcal{M}$  be the message space):*

- $\text{Setup}(1^\lambda)$ : output the master secret key  $\text{MSK}$ .
- $\text{KeyGen}(\text{MSK}, f)$ : for  $f \in \mathcal{F}$ , output a function key  $\text{SK}_f$ .
- $\text{Enc}(\text{MSK}, m)$ : for  $m \in \mathcal{M}$ , output a ciphertext  $\text{CT}_m$ .
- $\text{Dec}(\text{SK}_f, \text{CT}_m)$ : output the function value  $f(m)$ .

FE is correct if for any  $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ ,  $\text{SK}_f \leftarrow \text{KeyGen}(\text{MSK}, f)$ ,  $\text{CT}_m \leftarrow \text{Enc}(\text{MSK}, m)$ , the probability of  $\text{Dec}(\text{SK}_f, \text{CT}_m) \neq f(m)$  is negligible.

Next, we define the (message-private) pIND-based security and function-private pIND-based security for private-key FE schemes.

**Definition 4.2.** *Given message space  $\mathcal{M}$  and function space  $\mathcal{F}$ , a  $q$ -CT (or unbounded-CT),  $Q$ -key (or unbounded-key) adaptive (message-private) pIND-CPA-security game for a private-key FE scheme is defined as the following:*

- *Setup*: The challenger runs  $\text{Setup}(1^\lambda)$  to get  $\text{MSK}$ , and randomly samples a bit  $b \leftarrow \{0, 1\}$ .

– *Query Phase:* The adversary can adaptively makes the following two types of queries:

- *Key Query:* The adversary chooses a p.p.t. sampling algorithm  $F$  which output is in  $\mathcal{F}$ , and gives it to the challenger. The challenger samples  $f \leftarrow F$ , generates  $sk_f = \text{KeyGen}(\text{MSK}, f)$  and returns  $f, sk_f$  to the adversary. This can be repeated adaptively for any polynomial times.
- *Ciphertext Query:* The adversary chooses two p.p.t. sampling algorithms  $M_0, M_1$  which outputs are in  $\mathcal{M}$  and gives them to the challenger. The challenger randomly chooses  $m \leftarrow M_b$ , generates  $ct_m \leftarrow \text{Enc}(\text{PK}, m)$  and returns  $ct_m$  to the adversary.

The number of key queries is bounded by  $Q$  or unbounded; the number of ciphertext queries is bounded by  $q$  or unbounded.

– *Output:* The adversary outputs  $b'$ , and the winning advantage for the adversary is defined by  $\text{Adv}^{\text{pIND}}(\mathcal{A}) = |\Pr(b' = b) - 1/2|$ .

Let  $q, Q$  be the number of ciphertext queries and key queries, we write the  $i$ -th key query and the chosen function by  $F^i, f^i$ , the  $j$ -th ciphertext query and the chosen message by  $M_0^j, M_1^j, m^j$ .

An adversary  $\mathcal{A}$  is said to be admissible, if the two distributions  $(F^i, F^i(M_0^1), \dots, F^i(M_0^q))_{i \in [Q]}$  and  $(F^i, F^i(M_1^1), \dots, F^i(M_1^q))_{i \in [Q]}$  are strictly computationally indistinguishable. We say that skFE is a (message-private) pIND-secure private-key FE if for any p.p.t. admissible adversary  $\mathcal{A}$ ,  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is negligible.

**Lemma 4.1.** For a (message-private) pIND adversary  $\mathcal{A}$  for a secret key FE scheme, define the trace of  $\mathcal{A}$  as:

$$\text{tr}_{\mathcal{A}} = ((M_0^j, M_1^j)_{j \in [q]}, (F^i, f^i, f^i(m^1), \dots, f^i(m^q))_{i \in [Q]}).$$

Then skFE is a (message-private) pIND-secure private-key FE, if and only if for every p.p.t.  $\mathcal{A}$  such that  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is non-negligible, there exists a p.p.t. algorithm  $\mathcal{T}$  which outputs the following distribution:

$$(aux, \bar{b}, (\bar{m}_j)_{j \in [q]}, \bar{tr} = ((\bar{M}_0^j, \bar{M}_1^j)_{j \in [q]}, (\bar{F}^i, \bar{f}^i, \bar{f}^i(\bar{m}^1), \dots, \bar{f}^i(\bar{m}^q))_{i \in [Q]})),$$

where  $\bar{b} \leftarrow \{0, 1\}, \bar{m}^j \leftarrow \bar{M}_{\bar{b}}^j$  for  $j \in [q], \bar{f}^i \leftarrow \bar{F}^i$  for  $i \in [Q]$ , and a p.p.t. algorithm  $\mathcal{B}$ , which satisfies:

- (1) For any p.p.t. algorithm  $\mathcal{P}$ ,  $\Pr(\mathcal{P}(\text{tr}_{\mathcal{A}}) = 1) - \Pr(\mathcal{P}(\bar{tr}) = 1)$  is negligible;
- (2)  $aux$  is independent with the following conditional distributions:  $\bar{m}^j | \bar{M}_0^j, \bar{M}_1^j, j \in [q]; \bar{f}^i | \bar{F}^i, i \in [Q]$ ;
- (3)  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is non-negligible.

*Proof.* The proof is similar to Lemma 3.1 and we omit the details. □

**Definition 4.3.** Given message space  $\mathcal{M}$  and function space  $\mathcal{F}$ , a  $q$ -CT (or unbounded-CT),  $Q$ -key (or unbounded-key) adaptive function-private pIND-CPA-security game for a private-key FE scheme is defined as the following:

- *Setup*: The challenger runs  $\text{Setup}(1^\lambda)$  to get  $\text{MSK}$ , and randomly samples a bit  $b \leftarrow \{0, 1\}$ .
  - *Query Phase*: The adversary can adaptively makes the following two types of queries:
    - *Key Query*: The adversary chooses two p.p.t. sampling algorithms  $F_0, F_1$  which output is in  $\mathcal{F}$ , and gives it to the challenger. The challenger samples  $f \leftarrow F_b$ , generates  $sk_f = \text{KeyGen}(\text{MSK}, f)$  and returns  $sk_f$  to the adversary. This can be repeated adaptively for any polynomial times.
    - *Ciphertext Query*: The adversary chooses two p.p.t. sampling algorithms  $M_0, M_1$  which outputs are in  $\mathcal{M}$  and gives them to the challenger. The challenger randomly chooses  $m \leftarrow M_b$ , generates  $ct_m \leftarrow \text{Enc}(\text{PK}, m)$  and returns  $ct_m$  to the adversary.
- The number of key queries is bounded by  $Q$  or unbounded; the number of ciphertext queries is bounded by  $q$  or unbounded.
- *Output*: The adversary outputs  $b'$ , and the winning advantage for the adversary is defined by  $\text{Adv}^{\text{pIND}}(\mathcal{A}) = |\Pr(b' = b) - 1/2|$ .

Let  $q, Q$  be the number of ciphertext queries and key queries, we write the  $i$ -th key query and the chosen function by  $F_0^i, F_1^i, f^i$ , the  $j$ -th ciphertext query and the chosen message by  $M_0^j, M_1^j, m^j$ .

An adversary  $\mathcal{A}$  is said to be admissible, if the two distributions  $(F_0^i(M_0^1), \dots, F_0^i(M_0^q))_{i \in [Q]}$  and  $(F_1^i(M_1^1), \dots, F_1^i(M_1^q))_{i \in [Q]}$  are strictly computationally indistinguishable. We say that skFE is a function-private pIND-secure private-key FE if for any p.p.t. admissible adversary  $\mathcal{A}$ ,  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is negligible.

**Lemma 4.2.** For a function-private pIND adversary  $\mathcal{A}$  for a secret key FE scheme, define the trace of  $\mathcal{A}$  as:

$$\text{tr}_{\mathcal{A}} = ((M_0^j, M_1^j)_{j \in [q]}, (F_0^i, F_1^i, f^i(m^1), \dots, f^i(m^q))_{i \in [Q]}).$$

Then skFE is a function-private pIND-secure private-key FE, if and only if for every p.p.t.  $\mathcal{A}$  such that  $\text{Adv}^{\text{pIND}}(\mathcal{A})$  is non-negligible, there exists a p.p.t. algorithm  $\mathcal{T}$  which outputs the following distribution:

$$(aux, \bar{b}, (\bar{m}^j)_{j \in [q]}, \bar{tr} = ((\bar{M}_0^j, \bar{M}_1^j)_{j \in [q]}, (\bar{F}_0^i, \bar{F}_1^i, \bar{f}^i(\bar{m}^1), \dots, \bar{f}^i(\bar{m}^q))_{i \in [Q]})),$$

where  $\bar{b} \leftarrow \{0, 1\}$ ,  $\bar{m}^j \leftarrow \bar{M}_b^j$  for  $j \in [q]$ ,  $\bar{f}^i \leftarrow \bar{F}_b^i$  for  $i \in [Q]$ , and a p.p.t. algorithm  $\mathcal{B}$ , which satisfies:

- (1) For any p.p.t. algorithm  $\mathcal{S}$ ,  $\Pr(\mathcal{S}(\text{tr}_{\mathcal{A}}) = 1) - \Pr(\mathcal{S}(\bar{tr}) = 1)$  is negligible;
- (2)  $aux$  is independent with the following conditional distributions:  $\bar{m}^j | \bar{M}_0^j, \bar{M}_1^j$ ,  $j \in [q]$ ;  $\bar{f}^i | \bar{F}_0^i, \bar{F}_1^i$ ,  $i \in [Q]$ ;
- (3)  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is non-negligible.

*Proof.* The proof is similar to Lemma 3.1 and we omit the details.  $\square$

We give a lemma on the existence of private-key pIND-secure FE.

**Lemma 4.3.** *There exists a private-key pIND-secure FE with 1-CT and unbounded key for P/poly, assuming the existence of one-way functions.*

*Proof.* By Theorem 3.1 (which can also be applied to private-key FE schemes), we can show that the SIM-secure private-key FE scheme for P/poly with 1-key and unbounded-CT queries in [GVW12] is also pIND-secure. If we can lift this scheme into a function-private pIND-secure private-key FE scheme, we can simply swap the KeyGen and Enc algorithms to obtain a private-key pIND-secure FE with unbounded-key and 1-CT for P/poly.

The lifting is similar to the one in [BS15]. Let skFE be the pIND-secure message-private private-key FE scheme, Sym be a symmetric encryption scheme, PRF be a pseudo-random function family. We construct the pIND-secure function-private private-key FE as follows:

- Setup( $1^\lambda$ ): Generate three symmetric encryption keys  $k, k', k'' \leftarrow \text{Sym.KeyGen}(1^\lambda)$ , let  $\text{skFE.MSK} \leftarrow \text{skFE.Setup}(1^\lambda)$ . Return  $\text{MSK} = (k, k', k'', \text{skFE.MSK})$ .
- KeyGen( $\text{MSK}, f$ ): Let  $\tilde{f}$  be defined as:  $\tilde{f}(m, r) = f(m)$ . Let  $c = \text{Sym.Enc}(k, f)$ ,  $c' = \text{Sym.Enc}(k', \tilde{f})$ ,  $c'' = \text{Sym.Enc}(k'', \tilde{f})$ . Return  $\text{skFE.KeyGen}(\text{skFE.MSK}, g_{c,c',c''})$ , where for any  $c_1, c_2, c_3, g_{c_1,c_2,c_3}(m, k_1, k_2, k_3, r)$  is defined as follows:
  - If  $k_1 \neq \perp$ , let  $f \leftarrow \text{Sym.Dec}(k_1, c_1)$ , return  $f(m; r)$ .
  - Else if  $k_2 \neq \perp$ , let  $f \leftarrow \text{Sym.Dec}(k_2, c_2)$ , return  $f(m; r)$ .
  - Else if  $k_3 \neq \perp$ , let  $f \leftarrow \text{Sym.Dec}(k_3, c_3)$ , return  $f(m; r)$ .
  - Else return  $\perp$ .
- Enc( $\text{MSK}, m$ ): Sample a random seed  $r$  and return  $ct \leftarrow \text{skFE.Enc}(\text{skFE.MSK}, (m, k, \perp, \perp, r))$ .
- Dec( $sk, ct$ ): return  $\text{skFE.Dec}(sk, ct)$ .

Now we prove the security of the construction above through a hybrid of games.

Game 0 is the original game.

In Game 1, the challenger first samples a uniform random seed  $r^*$ , and for each ciphertext query, returns  $\text{skFE.Enc}(\text{MSK}, (m, k, \perp, \perp, r^*))$  instead of  $\text{skFE.Enc}(\text{MSK}, (m, k, \perp, \perp, r))$  for a freshly sampled  $r$ . Game 0 and Game 1 are indistinguishable from the pIND-security of skFE. (Note that the distribution of messages in different ciphertext queries share the same  $r^*$ .)

In Game 2, when the adversary makes a key query, instead of sampling  $f \leftarrow F_b$  using a random seed, the challenger samples two seeds  $s_0, s_1$ , and uses  $\text{PRF}(r^*, s_b)$  as the seed to sample  $f \leftarrow F_b$ . Game 1 and Game 2 are indistinguishable from the pseudorandomness of PRF.

In Game 3, for each key query, let  $\tilde{c}' = \text{Sym.Enc}(k', G_{F_0, s_0})$ ,  $\tilde{c}'' = \text{Sym.Enc}(k'', G_{F_1, s_1})$ , returns  $\text{skFE.KeyGen}(\text{MSK}, g_{c, \tilde{c}', \tilde{c}''})$  instead of  $\text{skFE.KeyGen}(\text{MSK}, g_{c, c', c''})$ , where  $G_{F_b, s_b}(m, r)$  is defined as:

- Sample  $f \leftarrow F_b$  using the seed  $\text{PRF}(r, s_b)$ ;
- Return  $f(m)$ .

Game 2 and Game 3 are indistinguishable from the security of  $\text{Sym}$ .

In Game 4, we change the ciphertext into  $\text{skFE.Enc}(\text{MSK}, m, \perp, k', \perp, r^*)$  for  $b = 0$  and  $\text{skFE.Enc}(\text{MSK}, m, \perp, \perp, k'', r^*)$  for  $b = 1$ . Since  $f_b(m) = G_{F_b}(m, r^*)$ , we can see that the trace for  $\text{skFE}$  is the same in Game 3 and Game 4, so Game 3 and Game 4 are indistinguishable from the security of  $\text{skFE}$ .

In Game 5, for each key query, let  $\tilde{c} = \text{Sym.Enc}(k, \perp)$ , returns  $\text{skFE.KeyGen}(\text{MSK}, g_{\tilde{c}, \tilde{c}', \tilde{c}''})$  instead of  $\text{skFE.KeyGen}(\text{MSK}, g_{c, \tilde{c}', \tilde{c}''})$ . Game 4 and Game 5 are indistinguishable from the security of  $\text{Sym}$ . Note that the function  $g_{\tilde{c}, \tilde{c}', \tilde{c}''}$  is the same for  $b = 0$  and  $b = 1$  in Game 5.

Now in Game 5, if  $\mathcal{A}$  is an adversary for the function-private scheme with non-negligible advantage, there is an adversary  $\mathcal{A}'$  which is an adversary for  $\text{skFE}$  with non-negligible advantage. By the  $\text{pIND}$ -based security of  $\text{skFE}$ , there exist a sampling algorithm  $\mathcal{T}'$  and an algorithm  $\mathcal{B}'$  with non-negligible advantage which satisfy Lemma 4.1. We write the output of  $\mathcal{T}'$  as:

$$(aux, \bar{b}', (\bar{m}^j)_{j \in [q]}, \bar{tr}' = ((\bar{M}'_0, \bar{M}'_1)_{j \in [q]}, (\bar{F}'^i, \bar{f}'^i, \bar{f}'^i(\bar{m}^1), \dots, \bar{f}'^i(\bar{m}^q))_{i \in [Q]})).$$

Since  $\bar{tr}'$  is indistinguishable with  $tr_{\mathcal{A}'}$ , so elements in both  $\bar{tr}'$  and  $tr_{\mathcal{A}'}$  has the same structure, so we can write  $\bar{m}^j = (\bar{m}^j, \perp, \bar{k}', \perp, \bar{r}^*)$  for  $\bar{b}' = 0$  and  $\bar{m}^j = (\bar{m}^j, \perp, \perp, \bar{k}'', \bar{r}^*)$  for  $\bar{b}' = 1$ ,  $\bar{f}'^i = g_{\tilde{c}, \tilde{c}', \tilde{c}''}$  where  $\tilde{c}, \tilde{c}', \tilde{c}''$  are  $\text{Sym}$  ciphertexts of  $\perp, G_{\bar{F}_0, \bar{s}_0}, G_{\bar{F}_1, \bar{s}_1}$  defined as above.

Without loss of generalization, we suppose that  $\bar{k}', \bar{k}''$  are contained in  $aux$ , since  $\bar{k}', \bar{k}''$  are predetermined and independent with the choice of either queried message or function.

Now we construct  $\mathcal{T}$  and  $\mathcal{B}$  from  $\mathcal{T}'$  and  $\mathcal{B}'$ .

$\mathcal{T}$  does the following:

- Call  $\mathcal{T}'$  to get  $\bar{h}', aux, \bar{b}', (\bar{m}^j)_{j \in [q]}, \bar{tr}'$ ;
- Extract  $\bar{M}'_0, \bar{M}'_1, \bar{m}^j$  from  $\bar{M}'^j_0, \bar{M}'^j_1, \bar{m}^j$ ,  $\bar{F}'_0, \bar{F}'_1$  from  $\bar{f}'^i$ ;
- Sample  $\bar{f}^i \leftarrow \bar{F}'^i_{\bar{b}'}$ ;
- Return  $aux, \bar{b}', (\bar{m}^j)_{j \in [q]}, \bar{tr} = ((\bar{M}'_0, \bar{M}'_1)_{j \in [q]}, (\bar{F}'_0, \bar{F}'_1, \bar{f}^i(\bar{m}^1), \dots, \bar{f}^i(\bar{m}^q))_{i \in [Q]}))$ .

$\mathcal{B}(aux, \bar{tr})$  does the following:

- For each  $\bar{M}'^j_b, j \in [q], b \in \{0, 1\}$ , sampling from  $\bar{M}'^j_b$  does the following:
  - Sample  $\bar{m} \leftarrow \bar{M}'^j_b$ ;
  - If  $j = 1$ , sample a random seed  $\bar{r}^*$ , otherwise use the same  $\bar{r}^*$  as in  $j' < j^2$ ;
  - If  $b = 0$ , return  $\bar{m}^j = (\bar{m}, \perp, \bar{k}', \perp, \bar{r}^*)$ , otherwise return  $\bar{m}^j = (\bar{m}, \perp, \perp, \bar{k}'', \bar{r}^*)$ .
- For each  $\bar{F}'^i_0$  and  $\bar{F}'^i_1$ , sampling from  $\bar{F}'^i$  does the following:

<sup>2</sup> Here we allows different distributions  $\bar{M}'^j_b$  to include the same randomness  $\bar{r}^*$ , which means that there is a shared inner state between these sampling algorithms. We note that  $\text{SIM}$ -secure FE implies  $\text{pIND}$ -secure FE even considering stateful ciphertext queries like this, so it will not affect the validity of the proof.

- Sample two random seeds  $\bar{s}_0, \bar{s}_1$ ;
  - Let  $G_{\bar{F}_0, \bar{s}_0}$  and  $G_{\bar{F}_1, \bar{s}_1}$  be defined as above, and  $\bar{c}' = \text{Sym.Enc}(\bar{k}', G_{\bar{F}_0, \bar{s}_0})$ ,  $\bar{c}'' = \text{Sym.Enc}(\bar{k}'', G_{\bar{F}_1, \bar{s}_1})$ ;
  - Return  $\bar{f}'^i = g_{\bar{c}, \bar{c}', \bar{c}''}$ .
- Call  $\mathcal{B}'(aux, ((\bar{M}'_0, \bar{M}'_1)_{j \in [q]}, (\bar{F}'^i; \bar{f}'^i, \bar{f}'^i(\bar{m}'^1), \dots, \bar{f}'^i(\bar{m}'^q))_{i \in [Q]}))$  to get the output.

It is not hard to see that  $\mathcal{B}$  calls  $\mathcal{B}'$  exactly with  $(aux, \bar{tr}')$ , where  $\bar{tr}'$  is defined as above, and we already know that  $\Pr(\mathcal{B}'(aux, \bar{tr}') = \bar{b}) - 1/2$  is non-negligible. So we successfully construct  $\mathcal{T}$  and  $\mathcal{B}$  satisfies Lemma 4.2. Thus the new scheme is a function-private pIND-secure private-key FE scheme.  $\square$

**Theorem 4.1.** *There exists a construction for ad-pIND-secure FE for P/poly from sel-IND-secure FE for P/poly assuming the existence of one-way functions.*

*Proof.* We simply write down the [ABSV15] construction here, and give a high level proof. The details are similar to the ad-IND-security proof in [ABSV15]. Given the following primitives:

- A sel-IND secure public-key FE scheme for P/poly Sel;
- An ad-pIND secure 1-CT private-key FE scheme for P/poly OneCT;
- A symmetric encryption scheme with pseudorandom ciphertexts Sym;
- A pseudorandom function family PRF.

The adaptive scheme Ad is constructed as follows:

- Setup( $1^\lambda$ ): Sample  $(\text{Sel.PK}, \text{Sel.MSK}) \leftarrow \text{Sel.Setup}(1^\lambda)$ , and return  $\text{PK} = \text{Sel.PK}$ ,  $\text{MSK} = \text{Sel.MSK}$ .
- KeyGen( $\text{MSK}, f$ ): Sample  $C_E \leftarrow \{0, 1\}^{l_1(\lambda)}$ ,  $\tau \leftarrow \{0, 1\}^{l_2(\lambda)}$ , return  $sk_f \leftarrow \text{Sel.KeyGen}(\text{Sel.MSK}, G_{f, C_E, r})$ ,  $G_{f, C_E, r}(\text{OneCT.MSK}, K, \text{Sym.K}, \beta)$  defined as follows:
  - If  $\beta = 1$ , output  $\text{Sym.Dec}(\text{Sym.K}, C_E)$ ;
  - Otherwise, output  $\text{OneCT.KeyGen}(\text{OneCT.MSK}, f; \text{PRF}_K(\tau))$ .
- Enc( $\text{PK}, m$ ): Output  $\text{CT} = (\text{CT}_0 \leftarrow \text{OneCT.Enc}(\text{OneCT.MSK}, m), \text{CT}_1 \leftarrow \text{Sel.Enc}(\text{Sel.MPK}, (\text{OneCT.MSK}, K, 0^\lambda, 0)))$ .
- Dec( $sk_f, \text{CT}$ ): Output  $\text{OneCT.Dec}(\text{Sel.Dec}(sk_f, \text{CT}_1), \text{CT}_0)$ .

The ad-pIND-security of this construction can be proved by a hybrid of games. Let Game 0 be the original pIND-CPA game.

In Game 1,  $C_E$  is replaced by  $\text{Sym.Enc}(\text{Sym.K}^*, sk_f \leftarrow \text{OneCT.KeyGen}(\text{OneCT.MSK}, f; \text{PRF}_K(\tau)))$  for random  $\text{Sym.K}^*$ . Game 0 and Game 1 are indistinguishable from the security of Sym.

In Game 2,  $\text{CT}_1$  is replaced by  $\text{Sel.Enc}(\text{Sel.MPK}, (0^\lambda, 0^\lambda, \text{Sym.K}^*, 1))$ . Since any adversary  $\mathcal{A}$  distinguishing Game 1 and Game 2 makes only deterministic ciphertext queries to Sel, we can see that Game 1 and Game 2 are indistinguishable from the IND-security of Sel.

In Game 3,  $\text{PRF}_K(\tau)$  is replaced by a truly random  $R$ . Game 2 and Game 3 are indistinguishable by the pseudorandomness of PRF.

We see that any adversary  $\mathcal{A}$  which has non-negligible advantage in Game 3 has also a non-negligible advantage in the  $\text{ad-pIND-CPA}$  game of OneCT. Then if OneCT is  $\text{ad-pIND-secure}$ , we can construct  $\mathcal{B}$  and the input distribution  $(h', aux, tr_{\mathcal{B}})$  for  $\mathcal{A}$  which satisfies Lemma 3.1, hence Ad is  $\text{ad-pIND-secure}$ .  $\square$

## 5 Application of pIND-secure FE: Hashing a Secret Value

Next, we introduce a specific application scenario, which can be constructed from pIND-secure FE. This application is inspired by Example 1.1, the counterexample for IND-based security. We show that how we can use pIND-secure FE to output the hash of a secret value. Like blind signature [Cha82], we name this new primitive “blind hash”. We first give its syntax, which is similar to the syntax of functional encryption.

**Definition 5.1.** *A blind hash system consists of the following algorithms:*

- $\text{Setup}(1^\lambda, 1^n, 1^k)$ : output the public key  $pk$  and the main secret key  $msk$ . We require that  $n \geq k$ .
- $\text{HashGen}(msk, h)$ : for a hash function  $h : \{0, 1\}^n \rightarrow \{0, 1\}^k$ , output its blinded version  $H$ .
- $\text{Enc}(pk, m)$ : output the encrypted message  $c$ .

The blind hash system is called correct, if for  $(pk, msk) \leftarrow \text{Setup}(\lambda)$ ,  $H \leftarrow \text{HashGen}(msk, h)$ ,  $c \leftarrow \text{Enc}(pk, m)$ , the probability of  $H(c) \neq h(m)$  is negligible.

In this definition, we restrict the input length of the hash function to be  $n$  instead of arbitrary length, in order for  $\text{Enc}$  to be well-defined. We can choose large enough  $n$ , and pad any string with length  $n' < n$  into a string of length  $n$ .

We require the one-wayness of a blind hash system.

**Definition 5.2.** *A blind hash system  $(\text{Setup}, \text{HashGen}, \text{Enc})$  is called one-way, if for any p.p.t. adversary  $\mathcal{A}$  and a set  $\mathcal{S}$  of (polynomial number of) universal one-way hash families, the winning advantage of the following game is negligible:*

- *Setup*: The challenger runs  $\text{Setup}(1^\lambda)$  and returns  $pk$  to the adversary.
- *Phase 1*: Each time the adversary submits a universal one-way hash family  $\mathcal{H} \in \mathcal{S}$ , the challenger samples  $h \leftarrow \mathcal{H}$ , and returns  $(h, H \leftarrow \text{HashGen}(msk, h))$  to the adversary. This can be repeated for any polynomial numbers of times.
- *Challenge*: The challenger chooses  $m \leftarrow \mathcal{M}$ , and returns  $\text{Enc}(pk, m)$  to the adversary.
- *Phase 2*: Same as Phase 1.
- *Guess*: The adversary outputs  $m'$ . The winning advantage of  $\mathcal{A}$  is defined by  $\Pr(m' = m)$ .

In this definition, we give a set of universal one-way hash families outside the game instead of letting them to be chosen by the adversary, since both the adversary and the challenger are p.p.t., hence cannot have the ability to determine whether a hash family is universal one-way.

Before we give our construction for the blind hash system, we first introduce the Goldreich-Levin hardcore predicate for one-way functions.

**Definition 5.3.** *A polynomial time computable predicate  $b$  is a hardcore predicate of a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^k$ , if for any p.p.t. algorithm  $\mathcal{P}$ ,  $|\Pr_{m \leftarrow \{0, 1\}^n}(\mathcal{P}(f(m)) = b(m)) - 1/2|$  is negligible.*

**Lemma 5.1 (Goldreich-Levin Theorem).** *If  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a one-way function, then  $b(m, r) = \langle m, r \rangle$  is a predicate of the function  $g : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ ,  $g(m\|r) = f(m)\|r$ .*

Now we are ready to construct our blind hash system. Given a functional encryption scheme FE, the blind hash system is constructed as follows:

- **Setup**( $1^\lambda$ ): Run FE.Setup( $1^\lambda$ ) and output the public key  $pk$  and the main secret key  $msk$ .
- **HashGen**( $msk, h$ ): Let  $\bar{h}$  be the function which pads the output of  $h$  from  $k$  bits into  $n$  bits (by filling 0s). Let function  $g_h$  be defined as:  $g_h(m\|r) = \bar{h}(m)\|r$ . Calculate  $sk_{g_h} \leftarrow$  FE.KeyGen( $msk, g_h$ ). Let the blinded hash  $H(c)$  be defined as:
  - Let  $t \leftarrow$  FE.Dec( $sk_{g_h}, c$ );
  - Output the first  $k$  bits of  $t$ .
- **Enc**( $pk, m$ ): Let  $r \leftarrow \{0, 1\}^n$ , output FE.Enc( $pk, m\|r$ ).

**Theorem 5.1.** *Let FE be pIND-based secure, then the construction above is a one-way blind hash system.*

*Proof.* Let  $\mathcal{G}_{\mathcal{H}}$  be the p.p.t. algorithm that first samples  $h \leftarrow \mathcal{H}$  and then outputs  $g_h$  (as define above), and  $M_0$  (resp.  $M_1$ ) be a p.p.t. algorithm that outputs a random string  $m\|r \in \{0, 1\}^{2n}$  where  $\langle m, r \rangle = 0$  (resp.  $\langle m, r \rangle = 1$ ). For each adversary  $\mathcal{A}'$  attacks the one-wayness of the blind hash system, we consider any pIND adversary  $\mathcal{A}$  for FE which makes specific queries as follows:

- When  $\mathcal{A}'$  submits a query  $\mathcal{H}_j$  in Phase 1 or Phase 2,  $\mathcal{A}$  submits  $\mathcal{G}_{\mathcal{H}_j}$ , and gets  $sk_{g_h}$  from inside the blinded hash function  $H$ .
- At the challenge phase,  $\mathcal{A}$  submits  $(M_0, M_1)$  to the challenger, and gets the challenge ciphertext of  $\mathcal{A}'$ .

We do not restrict the way that  $\mathcal{A}$  gives its outputs  $b'$ .

By the definition of pIND-based security, if  $Adv(\mathcal{A})$  is non-negligible, there exists a sampling algorithm  $\mathcal{T}$  and an algorithm  $\mathcal{B}$ , where  $(aux, \bar{b}, \bar{m}, \bar{tr}) \leftarrow \mathcal{T}$ ,  $\bar{tr}$  is computationally indistinguishable from  $tr_{\mathcal{A}}$ , and  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is non-negligible, where  $\bar{tr}$  takes the form as:

$$\bar{tr} = (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}))_{i \in [Q]}).$$



Since  $M_0, M_1$  are fixed and each  $F_i$  in  $tr_{\mathcal{A}}$  is chosen only from a pre-determined polynomial size set  $\{\mathcal{G}_{\mathcal{H}}\}_{\mathcal{H} \in \mathcal{S}}$ , we can see that the computational indistinguishability between  $tr_{\mathcal{A}}$  and  $\bar{tr}$  implies that  $\bar{M}_0 = M_0, \bar{M}_1 = M_1$ , and  $\bar{F}_i = \mathcal{G}_{\mathcal{H}}$  for some  $\mathcal{H} \in \mathcal{S}$ . We also write  $\bar{f}_i$  as  $g_{\bar{h}_i}$  where  $\bar{h}_i \in \mathcal{H}$ , thus  $\bar{f}_i(\bar{m}) = g_{\bar{h}_i}(m\|r)$  for some  $m, r$ , and  $\bar{b} = \langle m, r \rangle$ .

Since  $aux$  is independent with the choice of  $f_i$  and  $\bar{m}$ , we define  $\mathcal{B}_i(g_{\bar{h}_i}(m\|r)) := \mathcal{B}_i(aux, \bar{tr})$ , so by a standard hybrid argument,  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is non-negligible, only if there exists  $\mathcal{B}_i$ , such that  $\Pr(\mathcal{B}_i(\{g_{\bar{h}_i}(m\|r)\}_{i \in [q]}) = \langle m, r \rangle) - 1/2$  is non-negligible. However, from Goldreich-Levin Theorem,  $\langle m, r \rangle$  is a hardcore predicate for  $g_{\bar{h}_i}(m\|r)$ , and since each  $\bar{h}_i$  is independently chosen from universal hash families, we have that  $\{g_{\bar{h}_i}(m\|r)\}_{i \in [q]}$  are independent, so  $\langle m, r \rangle$  is also a hardcore predicate for  $g(m\|r) := g_{\bar{h}_1}(m\|r) \parallel \dots \parallel g_{\bar{h}_q}(m\|r)$ , which means that  $\Pr(\mathcal{B}'(\{g_{\bar{h}_i}(m\|r)\}_{i \in [q]}) = \langle m, r \rangle) - 1/2$  must be negligible. So we have that  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is also negligible, hence  $Adv(\mathcal{A})$  is negligible.

We know that if a function is one-one, then having a hardcore predicate implies one-wayness. Since the advantage of  $\mathcal{A}$  is negligible, if we consider the function  $f(m\|r) = sk_{h_1} \parallel \dots \parallel sk_{h_q} \parallel ct, ct \leftarrow \text{FE.Enc}(pk, m\|r)$ , which is a one-one function, we see that  $\langle m, r \rangle$  is also its hardcore predicate, so  $f(m\|r)$  is one-way. Since  $r$  can be directly generated from  $\text{FE.Dec}(sk_i, ct)$  given any  $sk_i$ , we can see that  $f(m\|r)$  is one-way according to the input  $m$ , hence the advantage for  $\mathcal{A}'$  is also negligible. Thus we finish the proof.  $\square$

The construction from pIND-secure FE to blind hash systems are quite straightforward. Since pIND-secure FE can be constructed from IND-secure FE schemes, blind hash systems can be constructed from IND-secure FE schemes.

However, we show that the same method in this section cannot be used to directly construct blind hash systems from IND-secure FE: let  $h$  be a collision-resistant hash function, and construct the hash family  $\mathcal{H}$  be:  $\{h_k : h_k(m) := h(k\|m)\}$ . So if  $\mathcal{A}$  make an admissible query, which means that  $h_{k_1}(m) = h_{k_2}(m)$ , it finds a collision for  $h$ , which contradicts the security of  $h$ , so  $\mathcal{A}$  cannot make any queries. So if the construction above uses an IND-based FE scheme, the one-way property cannot be satisfied, like what we showed in Example 1.1.

Also, For SIM-based secure FE schemes, as it was proven in [AGVW13], there is no unbound-key SIM-based secure FE schemes supporting one-way functions, so SIM-based FE schemes for  $\mathcal{H}$  cannot be constructed, hence it is impossible to directly construct blind hash systems from SIM-based FE schemes.

## 6 Application of pIND-secure FE: Semi-universal Proxy Re-encryption

Now we give another application scenario which can be constructed from pIND-secure FE but not other security definitions. We consider proxy re-encryption (PRE) schemes [BBS98], which can be used to transform a ciphertext encrypted under a delegator key into one encrypted under a delegatee key, without leaking the plaintext. However, in most existing PRE constructions, the delegator encryption scheme and the delegatee encryption scheme must be the same:

they cannot re-encrypt a given ciphertext into another ciphertext under another public-key encryption scheme.

In [DN21], the authors introduced universal proxy re-encryption, and gave their construction from probabilistic  $i\mathcal{O}$ , where both the delegator and the delegatee can be arbitrary PKE schemes. Here, we discuss a weaker version of universal PRE, where only the delegatee ciphertext can be encrypted by arbitrary PKE schemes, and we call it semi-universal PRE. We now show that semi-universal PRE can be constructed by pIND-secure FE for P/poly. We note that pIND-secure FE for P/poly can be constructed from IND-secure FE for P/poly as we proved in Sect. 4, thus our construction of semi-universal PRE also has a weaker requirement than the existence of  $pi\mathcal{O}$  in the construction of universal PRE [DN21] (we note that even constructing  $i\mathcal{O}$  requires sub-exponential hardness IND-secure FE for P/poly).

We first give the syntax definition of semi-universal PRE.

**Definition 6.1.** *A semi-universal PRE consists of the following algorithms:*

- $\text{KeyGen}(1^\lambda)$ : *Output a public-key/secret-key pair  $(pk, sk)$ .*
- $\text{Enc}(pk, m)$ : *For a public key generated from  $\text{KeyGen}(1^\lambda)$ , output a ciphertext  $ct$  for  $m$ .*
- $\text{ReKeyGen}(sk_f, PKE, pk_t)$ : *Let  $sk_f$  be generated from  $\text{KeyGen}(1^\lambda)$  and  $pk_t$  be a public key of the PKE scheme  $PKE$ . This algorithm outputs a re-encryption key  $rk_{f \rightarrow t}$ .*
- $\text{ReEnc}(rk_{f \rightarrow t}, ct)$ : *Let  $ct$  be a ciphertext encrypted by  $pk_f$ , output a new ciphertext encrypted by  $pk_t$ .*
- $\text{Dec}(sk_f, ct)$ : *For a ciphertext  $ct \leftarrow \text{Enc}(pk_f, m)$ , output the corresponding message  $m$ .*

Let  $PKE = PKE.\text{KeyGen}, PKE.\text{Enc}, PKE.\text{Dec}$  be any public key encryption scheme. A semi-universal PRE scheme is correct, if for  $(pk_f, sk_f) \leftarrow \text{KeyGen}(1^\lambda)$ ,  $ct_f \leftarrow \text{Enc}(pk_f, m)$ , both: (1)  $\text{Dec}(sk_f, ct_f) = m$  except for a negligible probability; (2)  $(pk_t, sk, t) \leftarrow PKE.\text{KeyGen}(1^\lambda)$ ,  $rk_{f \rightarrow t} \leftarrow \text{ReKeyGen}(sk_f, PKE, pk_t)$ , the ciphertext  $ct_t \leftarrow \text{ReEnc}(rk_{f \rightarrow t}, ct_f)$  satisfies:  $PKE.\text{Dec}(sk_t, ct_t) = m$  except for a negligible probability.

We only define a weaker version of the single-hop security of PRE, where each delegator key must be generated at the setup phase, and allows only static corruption. For simplicity reason, we assume that the delegatee PKE scheme is always different from the delegator PKE scheme (which is a pIND-secure FE scheme as in our construction).

**Definition 6.2.** *For a semi-universal PRE  $(\text{Setup}, \text{Enc}, \text{ReKeyGen}, \text{ReEnc})$ , let  $\mathcal{P}$  be a set of semantic secure PKE scheme, and for any  $PKE \in \mathcal{P}$ ,  $\text{Enc} \neq PKE.\text{Enc}$ . The weak-CRA security of the semi-functional PRE is satisfied if for every adversary  $\mathcal{A}$ , the winning advantage of the following game is negligible:*

- *Setup*: The adversary asks the challenger to run  $\text{Setup}(1^\lambda)$  for any polynomial numbers of times to get  $(\widehat{pk}_i, \widehat{sk}_i)_{i \in [q]}$ . The challenger returns  $(\widehat{pk}_i)_{i \in [q]}$  to the adversary. Let  $L$  be an empty list.
  - *Phase 1*: The adversary can make one of the following types of queries in arbitrary sequence:
    - *Type 1*: The adversary submits  $\text{PKE} \in \mathcal{P}$ . The challenger generates  $(pk_{|L|+1}, sk_{|L|+1}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ , adds the pair  $(\text{PKE}; pk_{|L|+1})$  into  $L$ , and returns  $pk_{|L|+1}$  to the adversary.
    - *Type 2*: The adversary submits  $\text{PKE} \in \mathcal{P}$ . The challenger generates  $(pk_{|L|+1}, sk_{|L|+1}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ , adds the pair  $(\text{PKE}; pk_{|L|+1})$  into  $L$ , and returns  $(pk_{|L|+1}, sk_{|L|+1})$  to the adversary.
    - *Type 3*: The adversary submits  $\widehat{pk}_i, i \in [q]$ , and  $(\text{PKE}, pk_j) \in L$ . The challenger runs  $\text{ReKeyGen}(\widehat{sk}_i, \text{PKE}, pk_j)$  and returns  $rk_{i \rightarrow j}$  to the adversary if  $rk_{i \rightarrow j}$  has not been generated before.
- These queries can be repeated adaptively.
- *Challenge*: The adversary submits  $\widehat{pk}_{i^*}, i^* \in [q]$  and a pair of messages  $(m_0, m_1)$ , providing that for each Type 3 query which returns  $rk_{i^* \rightarrow j}$  for some  $j$ ,  $pk_j$  is generated from a Type 1 query. The challenger chooses  $b \leftarrow \{0, 1\}$ , and returns  $\text{Enc}(\widehat{pk}_{i^*}, m_b)$  to the adversary.
  - *Phase 2*: Same as Phase 1, under the restriction that for all Type 3 queries  $(\widehat{pk}_{i^*}, \text{PKE}, pk_j)$ ,  $pk_j$  is generated from a Type 1 query.
  - *Guess*: The adversary outputs  $b'$ . The winning advantage of  $\mathcal{A}$  is defined by  $|\text{Pr}(b' = b) - 1/2|$ .

Now we construct a weak-CRA secure semi-universal PRE from a pIND-secure functional encryption scheme FE. Let PRF be a pseudorandom function.

- $\text{KeyGen}(1^\lambda)$ : Output  $(pk, sk) \leftarrow \text{FE.Setup}(1^\lambda)$ .
- $\text{Enc}(pk, m)$ : Sample a random seed  $r$ , and output  $ct \leftarrow \text{FE.Enc}(pk, m \| r)$ .
- $\text{ReKeyGen}(sk_f, \text{PKE}, pk_t)$ : Sample a random key  $K$ , and let  $F(m \| r) := \text{PKE.Enc}(pk_t, m; \text{PRF}(K, r))$ . Return  $\text{FE.KeyGen}(sk_f, F)$ .
- $\text{ReEnc}(rk_{f \rightarrow t}, ct)$ : Output  $\text{FE.Dec}(rk_{f \rightarrow t}, ct)$ .
- $\text{Dec}(sk, ct)$ : Let  $sk_{ID} \leftarrow \text{FE.KeyGen}(sk, ID)$  where  $ID(m) = m$ , then output  $\text{FE.Dec}(sk_{ID}, ct)$ .

Before we prove the security of the PRE scheme, we first give a lemma to show that the auxiliary string has no effect in distinguishing a PKE ciphertext with random key.

**Lemma 6.1.** *Let PKE be a public key encryption scheme with semantic security. For a pair of messages  $m_0, m_1$ , any p.p.t. algorithm  $\mathcal{B}$  and auxiliary string  $aux$ , let  $(pk, sk) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ ,  $c_0 \leftarrow \text{PKE.Enc}(pk, m_0)$ ,  $c_1 \leftarrow \text{PKE.Enc}(pk, m_1)$ . Then  $\text{Pr}(\mathcal{B}(aux, pk, c_0) = 1) - \text{Pr}(\mathcal{B}(aux, pk, c_1) = 1)$  is negligible.*

*Proof.* Let  $\mathcal{B}_{aux}(\cdot, \cdot)$  be the algorithm  $\mathcal{B}(aux, \cdot, \cdot)$ . We construct a IND-CPA adversary  $\mathcal{A}$  for PKE, which submits  $m_0, m_1$  as the challenge messages, and runs

$\mathcal{B}_{aux}(pk, c)$  to get the output, then by the semantic security of PKE, the advantage of  $\mathcal{A}$  is negligible, hence  $\Pr(\mathcal{B}(aux, pk, c_0) = 1) - \Pr(\mathcal{B}(aux, pk, c_1) = 1)$  is negligible.  $\square$

We note that the adversary  $\mathcal{A}$  in the proof above is non-uniform, so the scheme PKE must be secure against non-uniform adversaries, which is a rather standard assumption.

**Theorem 6.1.** *Let FE be pIND-based secure, then the construction above satisfies weak-CRA security.*

*Proof.* Given an adversary  $\mathcal{A}'$  for the semi-universal PRE game. We construct a pIND adversary  $\mathcal{A}$  for FE as follows:

In the setup phase, suppose that the adversary asks the challenger to run  $\text{Setup}(1^\lambda)$  for  $q$  times.  $\mathcal{A}$  randomly choose  $i' \leftarrow [q]$ , and asks for the FE public key  $pk$ . Let  $\widehat{pk}_{i'} := pk$ . For  $i \neq i'$ , the challenger runs  $\text{FE.Setup}(1^\lambda)$  to get  $(\widehat{pk}_i, \widehat{sk}_i)$ .  $\mathcal{A}$  returns  $(\widehat{pk}_i)_{i \in [q]}$ .

When  $\mathcal{A}'$  generates a Type 1 query PKE, let  $F_{\text{PKE}}$  be the following algorithm:

- Run  $(\text{PKE}.pk, \text{PKE}.sk) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ ;
- Sample a random key  $K$  and return the function  $f$  where  $f(m||r) := \text{PKE.Enc}(\text{PKE}.pk, m; \text{PRF}(K, r))$ .

$\mathcal{A}$  submits a KeyGen query  $F_{\text{PKE}}$ , and gets  $(f, sk_f)$ , where  $f$  contains  $\text{PKE}.pk$ . Let  $pk_{|L|+1} = \text{PKE}.pk$ , store  $sk_{f|L|+1} := sk_f$ . Return  $pk_{|L|+1}$  and add  $(\text{PKE}, pk_{|L|+1})$  into  $L$ .

For a Type 2 query PKE, return  $(pk_{|L|+1}, sk_{|L|+1}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$  directly while adding  $(\text{PKE}, pk_{|L|+1})$  into  $L$ .

For a Type 3 query  $(\widehat{pk}_i, \text{PKE}, pk_j)$ , if  $i = i'$  and  $pk_j$  is generated from Type 2 queries, then return a random guess  $b' \leftarrow \{0, 1\}$  and abort. If  $i = i'$  and  $pk_j$  is generated from Type 1 queries, return  $rk_{i \rightarrow j} := sk_{f_j}$  (generated in Type 1 queries). If  $i \neq i'$ , return  $rk_{i \rightarrow j} \leftarrow \text{FE.KeyGen}(\widehat{sk}_i, f_j)$ .

In the challenge phase, if  $\mathcal{A}'$  queries for  $i^* \neq i'$ , then return a random guess  $b' \leftarrow \{0, 1\}$  and abort. Otherwise, let  $M_b, b \in \{0, 1\}$  be the algorithm that first randomly samples  $r$  and returns  $m_b||r$ . Submit  $(M_0, M_1)$  and get the ciphertext  $ct$ , return  $ct$  to  $\mathcal{A}'$ .

Finally, return the guess  $b'$  from  $\mathcal{A}'$ .

We can see that  $\mathcal{A}$  does not abort if and only if  $i^* = i'$ . Since  $q$  is polynomial, the non-aborting probability  $1/q$  is non-negligible, so if the advantage of  $\mathcal{A}'$  is non-negligible, the advantage of  $\mathcal{A}$  is also non-negligible. By the definition of pIND-based security, there exists a sampling algorithm  $\mathcal{T}$  and an algorithm  $\mathcal{B}$  satisfies the definition. We write the output of  $\mathcal{T}$  as  $aux, \bar{b}, \bar{m}, \bar{tr} = (\bar{M}_0, \bar{M}_1, (\bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}))_{i \in [Q]})$ .

Since each key query of  $\mathcal{A}$  is from a polynomial size set  $\{F_{\text{PKE}} : \text{PKE} \in \mathcal{S}\}$  and a ciphertext query  $M_b$  samples  $m_b||r, b \leftarrow \{0, 1\}$ , we can see that as long as  $\bar{tr}$  is computationally indistinguishable from  $tr_{\mathcal{A}}, \bar{F}_i \in \{F_{\text{PKE}} : \text{PKE} \in \mathcal{S}\}$

and  $\bar{M}_0, \bar{M}_1$  samples  $\bar{m}_0 \| r, \bar{m}_1 \| r$  for fixed  $\bar{m}_0, \bar{m}_1$  and random  $r$ . So we rewrite  $\bar{f}_i(\bar{m})$  as  $\bar{f}_i(\bar{m}_{\bar{b}} \| r) = \text{PKE.Enc}(pk, \bar{m}_{\bar{b}}; \text{PRF}(K, r))$ , which is indistinguishable from  $\text{PKE.Enc}(pk, \bar{m}_{\bar{b}})$  by the pseudorandomness of PRF.

Since  $aux$  is independent from the choice of  $\bar{f}_i$ , it is also independent from the choice of  $pk$ , by Lemma 6.1, we have that  $\Pr(\mathcal{B}(aux, (\dots, \bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}_0 \| r), \dots)) = 1) - \Pr(\mathcal{B}(aux, (\dots, \bar{F}_i, \bar{f}_i, \bar{f}_i(\bar{m}_1 \| r), \dots)) = 1)$  is negligible. By a standard hybrid argument, we have that  $\Pr(\mathcal{B}(aux, \bar{tr}) = 1 | \bar{b} = 0) - \Pr(\mathcal{B}(aux, \bar{tr}) = 1 | \bar{b} = 1)$  is negligible, hence  $\Pr(\mathcal{B}(aux, \bar{tr}) = \bar{b}) - 1/2$  is negligible, which makes a contradiction. So the advantage of  $\mathcal{A}'$  is negligible, thus we finish the proof.  $\square$

By a discussion similar to Sect. 5, we can see that SIM-based and IND-based FE schemes cannot be used to construct semi-universal PRE schemes directly. We also point out that why semi-universal PRE cannot be directly constructed from rFE [GJKS15]. The SIM-based secure rFE in [GJKS15] supports only selective security, hence cannot satisfy our security definition. (We note that adaptively SIM-based secure rFE also suffers from the impossible result of [AGVW13].) For IND-based secure rFE, the authors require that each post-challenge key query  $f$ , where  $f$  is a probabilistic function, satisfies that  $f(m_0)$  and  $f(m_1)$  are statically indistinguishable, rather than computationally indistinguishable, hence cannot be satisfied if  $m_0 \neq m_1$  and  $f$  is  $\text{PKE.Enc}(pk, \cdot)$  for a PKE scheme PKE. Even if we consider only pre-challenge key queries, where the authors only require that  $f(m_0)$  and  $f(m_1)$  are computationally indistinguishable, it still cannot handle the case where  $f$  is  $\text{PKE.Enc}(pk, \cdot)$  since the adversary may hold the secret key  $sk$  corresponding to  $pk$ . The same thing happens for the distributional indistinguishability definition [AM18], which also requires  $f(m_0)$  and  $f(m_1)$  to be computationally indistinguishable.

## 7 Conclusion and Future Works

In this paper, we define a new security notion for FE: indistinguishability-based security against probabilistic queries (pIND-security). We justify our security notion from the following four points: (1) Our pIND-security is strictly between the classical SIM-security and IND-security; (2) Our pIND-security has both 1-CT to many-CT and selective to adaptive reductions; (3) We give a construction of fully secure FE for P/poly which satisfies pIND-security; (4) We give applications that can be directly constructed from pIND-secure FE schemes, but cannot be constructed from SIM-secure or IND-secure FE schemes in a same way.

We believe that our new definition has more potential applications than what we showed in this paper. We also hope that this new security notion can be used to simplify the construction from FE to  $i\mathcal{O}$ , hence pushing  $i\mathcal{O}$  further into practical.

**Acknowledgement.** This work is partially supported by the National Key R&D Program of China (No. 2020YFA0712300) and the National Natural Science Foundation of China (No. 62202294, No. 62272294).

## References

- [ABSV15] Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48000-7\\_32](https://doi.org/10.1007/978-3-662-48000-7_32)
- [AGVW13] 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](https://doi.org/10.1007/978-3-642-40084-1_28)
- [AJ15] Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-47989-6\\_15](https://doi.org/10.1007/978-3-662-47989-6_15)
- [AKW18] Agrawal, S., Koppula, V., Waters, B.: Impossibility of simulation secure functional encryption even with random oracles. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11239, pp. 659–688. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03807-6\\_24](https://doi.org/10.1007/978-3-030-03807-6_24)
- [ALMT20] Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 34–64. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45374-9\\_2](https://doi.org/10.1007/978-3-030-45374-9_2)
- [AM18] 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](https://doi.org/10.1007/978-3-030-03810-6_18)
- [AW17] Agrawal, S., Wu, D.J.: Functional encryption: deterministic to randomized functions from simple assumptions. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 30–61. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56614-6\\_2](https://doi.org/10.1007/978-3-319-56614-6_2)
- [BB04] Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24676-3\\_14](https://doi.org/10.1007/978-3-540-24676-3_14)
- [BCJ+19] Bartusek, J., et al.: Public-key function-private hidden vector encryption (and more). In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 489–519. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34618-8\\_17](https://doi.org/10.1007/978-3-030-34618-8_17)
- [BF13] Barbosa, M., Farshim, P.: On the semantic security of functional encryption schemes. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 143–161. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36362-7\\_10](https://doi.org/10.1007/978-3-642-36362-7_10)
- [BO13] Bellare, M., O’Neill, A.: Semantically-secure functional encryption: possibility results, impossibility results and the quest for a general definition. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 218–234. Springer, Cham (2013). [https://doi.org/10.1007/978-3-319-02937-5\\_12](https://doi.org/10.1007/978-3-319-02937-5_12)

- [BRS13] Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40084-1\\_26](https://doi.org/10.1007/978-3-642-40084-1_26)
- [BS15] Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46497-7\\_12](https://doi.org/10.1007/978-3-662-46497-7_12)
- [BSW11] 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](https://doi.org/10.1007/978-3-642-19571-6_16)
- [Cha82] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T., eds., *Advances in Cryptology: Proceedings of CRYPTO '82*, Santa Barbara, California, USA, August 23–25, 1982, pp. 199–203. Plenum Press, New York (1982)
- [DN21] Döttling, N., Nishimaki, R.: Universal proxy re-encryption. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12710, pp. 512–542. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-75245-3\\_19](https://doi.org/10.1007/978-3-030-75245-3_19)
- [GGH+13] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26–29 October, 2013, Berkeley, CA, USA, pp. 40–49. IEEE Computer Society (2013)
- [GHRW14] Gentry, C., Halevi, S., Raykova, M., Wichs, D.: Outsourcing private RAM computation. In: 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18–21, 2014, pp. 404–413. IEEE Computer Society (2014)
- [GJKS15] Goyal, V., Jain, A., Koppula, V., Sahai, A.: Functional encryption for randomized functionalities. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 325–351. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46497-7\\_13](https://doi.org/10.1007/978-3-662-46497-7_13)
- [GP21] Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: Samir Khuller and Virginia Vassilevska Williams, editors, STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021, pp. 736–749. ACM (2021)
- [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](https://doi.org/10.1007/978-3-642-32009-5_11)
- [JLS21] Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Khuller, S., Williams, V.V., editors, STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021, pp. 60–73. ACM (2021)
- [KLM+18] 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](https://doi.org/10.1007/978-3-319-98113-0_29)






- [LPST16] Lin, H., Pass, R., Seth, K., Telang, S.: Output-compressing randomized encodings and applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 96–124. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49096-9\\_5](https://doi.org/10.1007/978-3-662-49096-9_5)
- [MSH+19] Marc, T., Stopar, M., Hartman, J., Bizjak, M., Modic, J.: Privacy-enhanced machine learning with functional encryption. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11735, pp. 3–21. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-29959-0\\_1](https://doi.org/10.1007/978-3-030-29959-0_1)
- [O’N10] O’Neill, A.: Definitional issues in functional encryption. In: IACR Cryptol. ePrint Arch., p. 556 (2010)
- [PMR19] Patrnanabis, S., Mukhopadhyay, D., Ramanna, S.C.: Function private predicate encryption for low min-entropy predicates. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 189–219. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17259-6\\_7](https://doi.org/10.1007/978-3-030-17259-6_7)
- [RSG+19] Ryffel, T., Sans, E.D., Gay, R., Bach, F.R., Pointcheval, D.: Partially encrypted machine learning using functional encryption. CoRR, abs/1905.10214 (2019)
- [WW21] Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 127–156. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77883-5\\_5](https://doi.org/10.1007/978-3-030-77883-5_5)



**ZK I**



# A Generic Transform from Multi-round Interactive Proof to NIZK

Pierre-Alain Fouque<sup>1</sup> , Adela Georgescu<sup>2</sup> , Chen Qian<sup>3,4</sup> ,  
Adeline Roux-Langlois<sup>5</sup> , and Weiqiang Wen<sup>6</sup> 

<sup>1</sup> Rennes University, CNRS, INRIA, Rennes, France

<sup>2</sup> Department of Computer Science, University of Bucharest, Bucharest, Romania

<sup>3</sup> Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, China  
chen.qian@sdu.edu.cn

<sup>4</sup> School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China

<sup>5</sup> Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

<sup>6</sup> LTCI, Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France  
weiqiang.wen@telecom-paris.fr

**Abstract.** We present a new generic transform that takes a multi-round interactive proof for the membership of a language  $\mathcal{L}$  and outputs a non-interactive zero-knowledge proof (not of knowledge) in the common reference string model. Similar to the Fiat-Shamir transform, it requires a hash function  $H$ . However, in our transform the zero-knowledge property is in the standard model, and the adaptive soundness is in the non-programmable random oracle model (NPROM). Behind this new generic transform, we build a new generic OR-composition of two multi-round interactive proofs. Note that the two common techniques for building OR-proofs (parallel OR-proof and sequential OR-proof) cannot be naturally extended to the multi-round setting. We also give a proof of security for our OR-proof in the quantum oracle model (QROM), surprisingly the security loss in QROM is independent from the number of rounds.

## 1 Introduction

Non-interactive zero-knowledge (NIZK) proofs [18, 25] can prove a statement without leaking any additional information about the witness. Since its first introduction, NIZK plays an important role in constructing almost every primitive from the basic ones like chosen-ciphertext encryption [33], signature [22] to complex cryptographic protocols like e-voting [17], and e-cash system [12].

**Fiat-Shamir and Random Oracle Model.** The most common and efficient way to construct a non-interactive zero-knowledge proof in the random oracle model (ROM) is via the Fiat-Shamir transform [22]. One first constructs a  $\Sigma$ -protocol (1-round interactive proof), then turns it into non-interactive by simulating the random challenge using a hash function modeled as a random oracle.

Since its first introduction [2], the random oracle model (ROM) has been controversial. The advantage of ROM is that, it is generally easier to build cryptographic

primitives with it, and the resulting primitives are usually more efficient than their standard model version (without random oracle). However, a decade after its introduction Canetti, Goldreich and Halevi [10] discovered that the instantiation of RO is theoretically impossible. More precisely, there exist cryptosystems that are secure in the random oracle model, but for which replacing the random oracle by any implementation leads to an insecure cryptosystem. Therefore, standard model constructions are usually considered as more secure than the constructions in ROM.

Beside of theoretical impossibility, ROM also suffers from some security concerns in real world applications. For example, a common way to instantiate the random oracle is with hash functions (like MD5, SHA-1, SHA-2, SHA-3 etc.). Therefore, any progress in cryptanalysis of hash functions could potentially make the ROM-based schemes insecure. As a concrete example, the work of [35, 38] have shown that standard hash functions like MD5 or SHA-1 are far from behaving like random oracles. Based on these attacks, Stevens *et al.* [36] showed an attack on constructing two colliding X.509 certificates for different identities and public keys, while the system is still secure in the ROM.

**NIZK Without Random Oracle.** Efficient NIZK in the standard model is considered as a challenging problem. In the ‘ NIZK in the standard model has been proposed by [26]. However, the situation of the efficient standard model NIZK in the post-quantum setting is less clear. Several works have constructed efficient post-quantum NIZK schemes by relaxing the soundness definition (only average-case soundness [14] against classical worst-case soundness) or the syntax of NIZK itself (Designated-Verifier NIZK [31], NIZK in the preprocessing model [28]). The full-fledged post-quantum NIZK in the standard model is only due to a new framework in the recent breakthrough results [8, 9], which gives the first lattice-based NIZK without RO [34]. As another instantiation of this framework, a new NIZK based on Learning Parity with Noise assumption and Trapdoor Hash Functions has also been proposed [6]. However, the efficiency of all these constructions in the standard model is still far from that of post-quantum NIZK in ROM [7, 21, 32].

**Non-Programmable Random Oracle.** In recent years, there is another research direction of NIZK consists of replacing the ROM by its weaker variant non-programmable random oracle model NPROM, while preserving the efficiency [15, 29]. These constructions are both generic transforms from  $\Sigma$ -protocols to NIZK. Interestingly, they both have zero-knowledge property in the standard model, and soundness property in the non-programmable random oracle model (NPROM).

Another interesting point about these two constructions is that, their zero-knowledge property is independent of the random oracle model. Therefore, in many applications, such as e-voting or authenticated encryptions, it guarantees that even the hash function is broken in the future, the privacy is still preserved.

**Limits of NIZK in NPROM.** One big problem of both transforms [15, 29] is that, they only work for  $\Sigma$  protocols but not the more generic multi-round public-coin interactive proofs (PCIP). As several recent results of interactive proofs are exploiting the multi-round property of PCIP to gain efficiency, such as bullet proofs [7], exact proofs [21] or amortized exact proofs [4], an interesting question would be to extend the [15, 29]

transforms to multi-round interactive protocols. Moreover, between these two transforms, [15] not only requires less properties of the starting  $\Sigma$ -protocol than [29] (optimal soundness against special soundness) but it is also more efficient. Therefore, we have chosen to focus on extending [15] in this paper. Unfortunately, it cannot be easily extended, as its principal building block is an OR-composition of two  $\Sigma$ -protocol, and the existing OR-composition techniques do not apply to multi-round PCIP. We will give below a quick overview of the existing OR-proofs.

**OR-Proof.** The OR-composition of  $\Sigma$ -protocols has been initially used to construct ring-signature schemes by [16] based on the programmable random oracle. Another OR-composition technique has been proposed by [1] to weaken the model, they only require the NPROM, and [1] has a shorter proof than [16] (one hash value less in the proof.) However, neither of them can be extended to the OR-composition of multi-round public-coin interactive proofs. Note that, for multi-round interactive proofs, we can firstly use Fiat-Shamir transform to reduce the number of rounds, then apply [16] or [1] to construct NIZK. But, the Fiat-Shamir transform requires programmability of the random oracle for the zero-knowledge property. As our goal is to keep the zero-knowledge property in the standard model, this approach does not work. This raises a natural question:

*Can we build a generic OR-composition of multi-round PCIP, with zero-knowledge in the standard model and soundness in NPROM?*

We will answer this question positively by giving a new technique for OR-composition.

**Security in the Quantum Random Oracle Model (QROM).** Security of random oracle model in the quantum setting is not a trivial problem. Intuitively, a quantum adversary can build the hash function and run the primitive himself by querying quantum states. Therefore, the adversary can get a superposition of exponentially many samples of the random oracle, which gives him more advantage than a classical adversary. Many recent works address this issue [19,20,30], and they give detailed analysis for the Fiat-Shamir transform in this setting. As we claim that we have a post-quantum zero-knowledge proof, we also give an analysis of our transform in the QROM.

## 1.1 Our Contributions

In this paper, we bring several contributions. Firstly, we propose a new generic transform from multi-round PCIP to NIZK, with zero-knowledge property in the standard model and soundness in NPROM. The principal new technique behind this transform is a new OR-proof of two different PCIPs. Surprisingly, the soundness in QROM of both multi-round PCIP to NIZK and OR-proof of PCIPs has a security loss of  $O(Q_H^4)$  which is independent from the number of rounds.

More precisely:

- We propose in this paper a new generic transform from multi-round public-coin interactive proofs (PCIP) to a non-interactive zero-knowledge proof system (NIZK). Compared to Fiat-Shamir transform, the zero-knowledge property of our transform is in the *standard model*, and soundness property is in the *non-programmable random oracle model* (NPROM) (RO without programmability). While comparing with similar type of transforms [15,29], ours additionally supports multi-round PCIP.
- Behind our generic transform, we have developed a new technique to generate an OR-proof from two optimal sound PCIP:  $\text{PCIP}_0, \text{PCIP}_1$ . The direct approach consists of using Fiat-Shamir transform to turn both  $\text{PCIP}_0$  and  $\text{PCIP}_1$  into  $\Sigma$ -protocols, then apply either [16] or [1] transform to get an OR-proof. Compared to the direct approach, the zero-knowledge property of our transform is in the *standard model*, and our adaptive soundness property is in the NPROM. We believe that this new OR-composition has other applications and independent interests.
- Finally, we analyze the soundness property of our OR-proof in the QROM. Note that the zero-knowledge property of our OR-proof is in the standard model, therefore it is naturally secure in the QROM. Moreover, our transform from PCIP to NIZK has the same security loss as our OR-proof. Surprisingly, the security loss of the soundness is  $O(Q_H^4)$  which is independent of the number of rounds.

## 1.2 Technical Overview

Our main technique consists in constructing the OR-proof for multi-round PCIPs. We dedicate this section to explain the intuition behind our OR-proof. Firstly, we will give a quick overview of the existing parallel OR-proof [16] and sequential OR-proof [1,24] as we will borrow ideas from both transforms. Then, we explain why they can not be extended to  $n$ -round PCIPs, and our new techniques of OR-proof.

**Why [16] Does Not Work for  $n$ -round PCIPs?** Given  $\Sigma_0$  and  $\Sigma_1$  two  $\Sigma$ -protocols with transcripts  $\{R_0, h_0, s_0\}$  and  $\{R_1, h_1, s_1\}$ , the intuition behind the parallel [16] transform is that, after generating the first round commitments  $(R_0, R_1)$ , the corresponding challenges are chosen such that  $h_0 \oplus h_1 = H(R_0, R_1)$ . Therefore any adversary can freely choose one (and only one) between  $h_0$  and  $h_1$  even before seeing  $(R_0, R_1)$ . By using the HVZK property of the  $\Sigma$ -protocol, once  $h_0$  (or  $h_1$ ) chosen, the adversary can simulate the proof  $(R_0, s_0)$  or  $(R_1, s_1)$  without knowing any witness.

Let us now see why this approach can not be extended to  $n$ -round interactive protocols when  $n > 1$ . The natural extension of [16] would be to define the  $i$ -th round challenges ( $i \in [n]$ ) such that  $h_{i,0} \oplus h_{i,1} = H(\{R_{j,0}, R_{j,1}\}_{j=1}^i)$ . This transform is not secure. To show this, we construct below an example of two 2-round protocols that are secure individually, but once combined, the resulting OR-proof is not secure anymore.

**Counter-Example of [16] Applying on 2-round PCIPs.** Given two  $\Sigma$ -protocols  $\Sigma_0$  and  $\Sigma_1$ , we will construct two 2-round protocols  $\text{PCIP}_0, \text{PCIP}_1$  by adding one unused round into each of  $\Sigma_0, \Sigma_1$  but in different order. Namely, valid transcripts of  $\text{PCIP}_0$  and  $\text{PCIP}_1$  are of the form  $(\bar{R}_0, \bar{h}_0, R_0, h_0, s_0)$  and  $(R_1, h_1, \bar{R}_1, \bar{h}_1, s_1)$ , where  $(\bar{R}_0, \bar{h}_0, \bar{R}_1, \bar{h}_1)$  are just random strings and ignored in the verification process. If we

apply the naive extension of [16] transform to  $\text{PCIP}_0$  and  $\text{PCIP}_1$ , an adversary  $\mathcal{A}$  can randomly choose  $h_0, h_1$ , then use HVZK to simulate  $(R_0, h_0, s_0)$  and  $(R_1, h_1, s_1)$ . As  $(\bar{R}_0, \bar{h}_0, \bar{R}_1, \bar{h}_1)$  are ignored by the individual verification of  $\text{PCIP}_0$  and  $\text{PCIP}_1$ ,  $\mathcal{A}$  can define  $\bar{R}_0, \bar{R}_1$  to be random strings and

$$\bar{h}_0 := h_1 \oplus H(\bar{R}_0, R_1), \quad \bar{h}_1 := h_0 \oplus H(R_0, \bar{R}_1).$$

By the correctness of  $\text{PCIP}_0$  and  $\text{PCIP}_1$ ,  $(\bar{R}_0, \bar{h}_0, R_0, h_0, s_0, R_1, h_1, \bar{R}_1, \bar{h}_1, s_1)$  is a valid proof for which  $\mathcal{A}$  does not need to know any witness in order to produce it, so he can easily break soundness of the OR-proof composition.

The above attack works because we have given too much "freedom" to  $\mathcal{A}$ . He can freely chose one challenge per round. Therefore, we need to limit  $\mathcal{A}$  to only be able to freely choose the challenges from the same interactive protocol.

**Overview of Sequential OR-Proof [1, 24].** Given two  $\Sigma$ -protocols  $\Sigma_0$  and  $\Sigma_1$ , together with two statements  $x_0, x_1$  and a witness  $w_0$ . (*w.l.o.g.* we can assume that we know  $w_0$ .) The intuition of the sequential OR-proof is that  $H(R_0)$  is used as the challenge  $h_1$  for  $\Sigma_1$  and  $H(R_1)$  is used as the challenge  $h_0$  for  $\Sigma_0$ . The honest generation of the proof is given as in Fig. 1.

```

Prove( $x_0, x_1, w_0$ ):
01  $(R_0, st_0) \xleftarrow{\$} \Sigma_0.\text{Prove}_1(x_0, w_0)$ 
02  $h_1 := H(R_0)$ 
03  $(R_1, s_1) \xleftarrow{\$} \Sigma_1.\text{Sim}(x_1, h_1)$ 
04  $h_0 := H(R_1)$ 
05  $s_0 \xleftarrow{\$} \Sigma_0.\text{Prove}_2(x_0, w_0, h_0, st_0)$ 
06 return  $(R_0, R_1, h_0, h_1, s_0, s_1)$ 
    
```

**Fig. 1.** Prove algorithm of sequential OR-proof

The intuition behind the sequential OR-proof is that, one can freely choose to generate  $R_0$  or  $R_1$  first. However, once chosen to generate  $R_b$  and  $h_{1-b}$  first, then  $h_b$  will be chosen independently from the value  $R_b$ . By the soundness of  $\Sigma_b$  without  $w_b$ , no PPT adversary can generate a valid transcript  $\text{Trans}_b = (R_b, h_b, s_b)$ .

For  $n$ -round PCIPs, we can notice that before the honest side ( $b$ ) has been executed until the  $(n - 1)$ th round, the simulation side  $(1 - b)$  doesn't have all the challenges, therefore even an honest prover with  $w_b$  cannot generate a valid proof when  $n > 1$ .

**Intuition Behind Our Approach.** Let us consider two  $n$ -round public-coin interactive proofs  $\text{PCIP}_0$  and  $\text{PCIP}_1$  for proving the membership of two languages  $\mathcal{L}_0$  and  $\mathcal{L}_1$ . For simplicity, we assume  $\text{PCIP}_0$  and  $\text{PCIP}_1$  have same number of rounds in this section. We will prove that  $x_0 \in \mathcal{L}_0$  or  $x_1 \in \mathcal{L}_1$  without revealing exactly which witness is used. Let  $\text{Trans}_0 = (\{R_{i,0}, h_{i,0}\}_{i=1}^n, s_0)$  and  $\text{Trans}_1 = (\{R_{i,1}, h_{i,1}\}_{i=1}^n, s_1)$  be two transcripts of  $\text{PCIP}_0$  and  $\text{PCIP}_1$  respectively.

Our starting point is the parallel OR-proof. To prevent the above attack against multi-round parallel OR-proof, our idea is to combine all the challenges of the same side together by an offset. Therefore, once the offset and the first  $i$  rounds commitments are fixed, the challenges are fixed. More precisely, for  $b \in \{0, 1\}$ , we denote by  $A_b = \{a_{1,b}, \dots, a_{n,b}\}$  two offsets, we could compute the challenges of the  $i$ -th round as follows,

$$h_{i,0} = H(\{R_{j,0}\}_{j=1}^i) + a_{i,0}, \quad h_{i,1} = H(\{R_{j,1}\}_{j=1}^i) + a_{i,1}. \quad (1)$$

Now, the challenges are all related. We emphasize the fact that the adversary can freely choose  $A_b$ , where  $b \in \{0, 1\}$ , is equivalent to be able to choose every challenge of  $b$  side.

The second step is to only allow the adversary to freely choose one and only one offset between  $A_0$  and  $A_1$ . To do this, we borrow the idea from the sequential or-proof by putting  $A_0$  and  $A_1$  into the hash of the opposite side. More precisely, we have

$$h_{i,0} = H(\{R_{j,0}\}_{j=1}^i, A_1) + a_{i,0}, \quad h_{i,1} = H(\{R_{j,1}\}_{j=1}^i, A_0) + a_{i,1}. \quad (2)$$

As in sequential OR-proof, the order of query  $A_0$  and  $A_1$  is crucial in our case. Namely, at least one of the two cases must happen:

- Before the RO query on  $(\{R_{j,0}\}_{j=1}^i, A_1)$ , there exists a query of the form  $(\cdot, A_0)$ .
- Before the RO query on  $(\{R_{j,1}\}_{j=1}^i, A_0)$ , there exists a query of the form  $(\cdot, A_1)$ .

This forces the adversary to choose  $A_0$  before having seen  $H(\{R_{j,0}\}_{j=1}^i, A_1)$  or  $A_1$  before having seen  $H(\{R_{j,1}\}_{j=1}^i, A_0)$ . We can use this property to reduce the adaptive soundness of our OR-proof to the optimal soundness of the underlying PCIPs.

**Security in the QROM.** In our QROM security proof, we apply the Measure-then-Reprogram 2.0 technique [19]. There is a price to pay for proving our transform in the QROM, that is we need the programmability of the random oracle. Moreover, if we want to prove our transform for round-by-round, we need to program the random oracle in every round, this will introduce an exponential security loss in the number of rounds. Therefore, we restrict our transform to only optimal-sound PCIPs, then we can prove our transform with only  $O(Q_H^4)$  security loss.

Note that, despite the fact that our OR-proof is a composition of two multi-round PCIPs, we only need to apply the Measure-then-Reprogram 2.0 technique on 2 entries. This is due to the fact that our OR-proof is not a proof of knowledge, but only a proof of membership, which is already useful in many applications such as voting schemes etc.

Therefore, we do not need all the entries to be able to extract the witness. This observation makes our security loss of the adaptive soundness as low as  $O(Q_H^4)$  in QROM, which is independent from the number of rounds  $n$ . Different from our result, [19] has considered the soundness with proof of knowledge (stronger than our adaptive soundness) of Fiat-Shamir transform and their security loss is  $O(Q_H^{2n})$ .

Very recently, there is a new semi-generic transformation [27] from PCIPs to non-interactive proofs in the QROM while achieving proof of knowledge. However it requires the prover’s response to be in linear form. As a comparison, our transformation is generic and does not impose any restriction on the prover’s response.

In comparison, Unruh’s transform [37] works for any  $\Sigma$ -protocol, but introduces a noticeable overhead depending on the size of the challenge set. In [13], Chen et al. extend Unruh’s framework for a 3-round protocol where the second challenge is binary.

## 2 Preliminaries

### 2.1 Notations

For  $n \in \mathbb{N}$ , let  $[n] = \{1, \dots, n\}$ . For a finite set  $\mathcal{S}$ , we denote the sampling of a uniform random element  $x$  by  $x \xleftarrow{\$} \mathcal{S}$ . For simplicity of the notations, we omit that every algorithm takes as input the public parameter  $\text{par}$ . For an algorithm  $A$  which takes  $x$  as input, we denote its computation by  $y \xleftarrow{\$} A(x)$ . We assume all the algorithms (including adversaries) in this paper to be probabilistic unless stated otherwise. We denote an algorithm  $A$  with access to an oracle  $\mathcal{O}$  by  $A^{\mathcal{O}}$ .

For an NP language  $\mathcal{L}$ , we denote by  $x \in_{\mathbf{w}} \mathcal{L}$  the fact that the statement  $x$  is in the language  $\mathcal{L}$  with the witness  $\mathbf{w}$ .

We use code-based games [3] to present our definitions and proofs. We implicitly assume all Boolean flags to be initialized to 0 (**false**), numerical variables to 0, sets to  $\emptyset$  and strings to  $\perp$ . We make the convention that a procedure terminates once it has returned an output.  $\text{Exp}_{\Sigma, A}^G(1^\lambda) = b$  denotes the final (Boolean) output  $b$  of the adversary  $A$  running the security experiment  $G$  on the scheme  $\Sigma$  with security parameter  $\lambda$ , and if  $b = 1$  we say  $A$  wins  $G$ . The randomness in  $\Pr[\text{Exp}_{\Sigma, A}^G(1^\lambda) = 1]$  is over all the random coins in experiment  $G$ . Within a procedure, “**abort**” means that we terminate the run of an adversary  $A$ .

### 2.2 $n$ -Round Public Coin Interactive Proof (PCIP)

The general structure of an  $n$ -round Public-Coin Interactive Proof of the form depicted in Fig. 2 is defined as follows.<sup>1</sup> Notice that for  $n = 1$ , PCIP is a  $\Sigma$ -protocol, and PCIP is also named as identification scheme in some literatures.

**Definition 1 ( $n$ -round Public-Coin Interactive Proof).** *Let  $\mathcal{L}$  be an NP language. To prove a statement  $x \in_{\mathbf{w}} \mathcal{L}$ , an  $n$ -round public-coin interactive proof consists of  $n + 2$  PPT stateful algorithms  $\text{PCIP} = (\{\text{Prove}_i\}_{i=1}^{n+1}, \text{Verif})$  with the following syntax:*

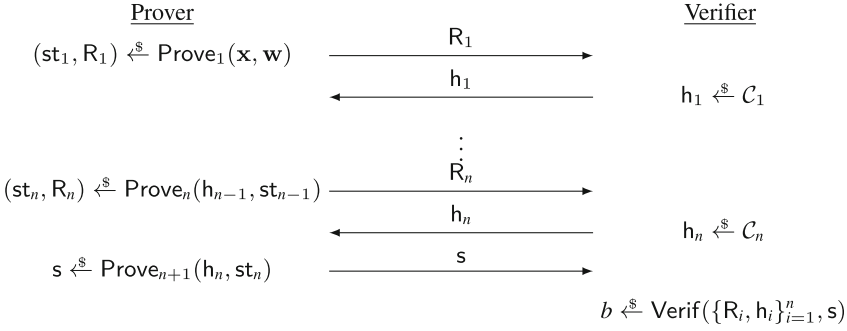
- $\text{Prove}_i(h_{i-1}, \text{st}_{i-1})$  takes a challenge  $h_{i-1}$  and a state  $\text{st}_{i-1}$  as input, and returns a commitment  $R_i$  and a new state  $\text{st}_i$ , where  $\text{st}_0 = (x, \mathbf{w})$ , and  $R_{n+1} = s$ .
- $\text{Verif}(x, (\{R_i, h_i\}_{i=1}^n, s))$ : The verification  $\text{Verif}$  takes as input a statement  $x$  and a transcript  $(\{R_i, h_i\}_{i=1}^n, s)$  and returns a decision 0 or 1.

We introduce the following definitions for a PCIP scheme:

---

<sup>1</sup> In this paper, we use the convention that  $n$ -round PCIP has  $2n + 1$  moves.





**Fig. 2.** An  $n$ -round Interactive Protocol

- **Transcript:** We define a **transcript** as all messages between the prover and the verifier of the form  $\text{Trans} = (\{R_i, h_i\}_{i=1}^n, s)$ . Moreover, we define a partial transcript  $\text{Trans}'$  as prefix of another transcript of the form  $(\{R_i, h_i\}_{i=1}^j)$  with  $j \leq n$ .

We require the following properties for an  $n$ -round PCIP:

- **Correctness:** For all  $(\mathbf{x}, \mathbf{w})$  such that  $\mathbf{x} \in_{\mathbf{w}} \mathcal{L}$  and for all honestly generated transcripts  $\text{Trans} = (\{R_i, h_i\}_{i=1}^n, s)$  using  $(\mathbf{x}, \mathbf{w})$ , we say that PCIP is  $\rho$ -correct if we have:

$$\Pr[\text{Verif}(\mathbf{x}, (\{R_i, h_i\}_{i=1}^n, s)) = 0] \leq \rho.$$

- **Honest-Verifier Zero-Knowledge:** For all  $(\mathbf{x}, \mathbf{w})$  such that  $\mathbf{x} \in_{\mathbf{w}} \mathcal{L}$ , we say that PCIP is  $\Delta$ -HVZK, if there exists a PPT simulator  $\text{Sim}$  that takes  $\mathbf{x}$  as input, and returns a transcript  $\text{Trans}$ , such that the distribution of  $\text{Trans}$  is at statistical distance at most  $\Delta$  from the distribution of an honestly generated transcript. In particular, if  $\Delta = 0$ , we say that PCIP has perfect HVZK.
- **Round-by-Round Soundness:** Let PCIP be an interactive-proof with  $i$ -th round challenge space  $\mathbb{Z}_{\ell_i}$ . We say that PCIP is round-by-round sound if, there exists a "doomed set"  $\mathcal{D} \in \{0, 1\}^*$  such that,
  - If  $\mathbf{x} \notin \mathcal{L}$ , then  $(\mathbf{x}, \emptyset) \in \mathcal{D}$ , where  $\emptyset$  denotes the empty transcript.
  - For all partial transcript  $\text{Trans}$ , such that  $(\mathbf{x}, \text{Trans}) \in \mathcal{D}$ , for all next message  $R_i$  given by the prover, there exists a negligible function  $\text{negl}(\cdot)$  such that

$$\Pr[(\mathbf{x}, \text{Trans} \| R_i \| h_i) \notin \mathcal{L} \mid h_i \stackrel{\$}{\leftarrow} \mathbb{Z}_{\ell_i}] \leq \text{negl}(\lambda).$$

- For any complete transcript  $\text{Trans}$ , if  $(\mathbf{x}, \text{Trans}) \in \mathcal{D}$  then  $\text{Verifier}(\mathbf{x}, \text{Trans}) = \text{false}$ .

Notice that the round-by-round soundness originally proposed by [8] is a very weak security notion. Since we only consider the constant rounds interactive proofs, by [8, Proposition 5.3 and 5.4] round-by-round soundness and negligible soundness are equivalent. On the other hand, optimal soundness (c.f. Definition 9 which is a multi-round version of special soundness) is a commonly used term for many protocols. If a protocol

is  $\varepsilon$ -optimal sound then it can be seen as no transcript can escape the doomed set except in one specific round with probability  $\varepsilon$ . Therefore, optimal soundness tightly implies round-by-round soundness. This provides us an alternative way to use our transform.

### 2.3 Non-Interactive Proof NIP

For the sake of completeness, we define two different types of non-interactive proofs NIP: Non-Interactive Zero-Knowledge proofs (NIZK) and Non-Interactive Witness Indistinguishable proofs (NIWI). Notice that we don't consider the proof of knowledge in this paper, and we use the adaptive soundness for NIPs.

**Definition 2 (Non-Interactive Proof NIP).** Let  $\mathcal{L}$  be an NP language. To prove a statement  $\mathbf{x} \in_{\mathbf{w}} \mathcal{L}$ , a non-interactive proof consists of four PPT algorithms  $\Pi = (\text{Setup}, \text{Prove}, \text{Verif}, \text{Sim} = (\text{Sim}_0, \text{Sim}_1))$  defined as follows:

- $\text{Setup}(1^\lambda) \rightarrow \text{CRS}$  : The setup algorithm Setup returns a common reference string CRS.
- $\text{Prove}(\text{CRS}, \mathbf{x}, \mathbf{w}) \rightarrow \pi$  : The prove algorithm Prove returns a proof  $\pi$  that  $\mathbf{x} \in_{\mathbf{w}} \mathcal{L}$  using  $\mathbf{w}$  as witness.
- $\text{Verif}(\text{CRS}, \mathbf{x}, \pi) \rightarrow \{0, 1\}$  : The verification algorithm Verif returns a decision, 1 (acceptance) or 0 (rejection).
- $\text{Sim}_0(1^\lambda) \rightarrow (\text{CRS}, \tau)$  : The first part of the simulation algorithm  $\text{Sim}_0$  outputs a common reference string CRS and a simulation trapdoor  $\tau$ .
- $\text{Sim}_1(\tau, \mathbf{x}) \rightarrow \pi$  : The second part of the simulation algorithm  $\text{Sim}_1$  outputs a simulated proof  $\pi$ .

We will also define the completeness, adaptive soundness, zero-knowledge, witness-indistinguishability of NIP as follows.

**Definition 3 ( $\rho$ -Completeness).** A NIP is  $\rho$ -complete if, for all  $\mathbf{x} \in_{\mathbf{w}} \mathcal{L}$  we have:

$$\Pr \left[ \text{Verif}(\text{CRS}, \mathbf{x}, \pi) = 0 \mid \begin{array}{l} \text{CRS} \xleftarrow{\$} \text{Setup}(1^\lambda) \\ \pi \xleftarrow{\$} \text{Prove}(\text{CRS}, \mathbf{x}, \mathbf{w}) \end{array} \right] \leq \rho.$$

**Definition 4 ( $(\varepsilon, Q_H)$ -Adaptive Soundness).** A NIP is  $(\varepsilon, Q_H)$ -adaptively sound in the non-programmable random oracle model NPROM, if for all PPT adversaries  $\mathcal{A}$  requiring at most  $Q_H$  hash queries we have:

$$\Pr \left[ \mathbf{x}^* \in \{0, 1\}^n \setminus \mathcal{L} \wedge \text{Verif}(\text{CRS}, \mathbf{x}^*, \pi^*) = 1 \mid \begin{array}{l} \text{CRS} \xleftarrow{\$} \text{Setup}(1^\lambda) \\ (\mathbf{x}^*, \pi^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\text{Hash}}}(\text{CRS}) \end{array} \right] \leq \varepsilon.$$

We consider the hash function as an NPRO in the soundness proof.

**Definition 5 (Zero-Knowledge).** A NIP is  $\Delta$ -Zero-Knowledge, if there exists a simulator  $\text{Sim} = (\text{Sim}_0, \text{Sim}_1)$  such that, the statistical distance between the output distributions of **Game** Sim and **Game** Real as defined in Fig. 3 is at most  $\Delta$ .

Moreover, if  $\Delta = 0$ , NIP is perfectly zero-knowledge.

<p><b>Game Sim:</b></p> <p>01 <math>(\text{CRS}, \tau) \xleftarrow{\\$} \text{Sim}_0(1^\lambda)</math></p> <p>02 <math>(\mathbf{x}, \mathbf{w}) \xleftarrow{\\$} \mathcal{A}(\text{CRS})</math>      <math>\  \mathbf{x} \in_{\mathbf{w}} \mathcal{L}</math></p> <p>03 <math>\pi \xleftarrow{\\$} \text{Sim}_1(\mathbf{x}, \tau)</math></p> <p>04 <b>return</b> <math>(\text{CRS}, \pi)</math></p>	<p><b>Game Real:</b></p> <p>05 <math>\text{CRS} \xleftarrow{\\$} \text{Setup}(1^\lambda)</math></p> <p>06 <math>(\mathbf{x}, \mathbf{w}) \xleftarrow{\\$} \mathcal{A}(\text{CRS})</math>      <math>\  \mathbf{x} \in_{\mathbf{w}} \mathcal{L}</math></p> <p>07 <math>\pi \xleftarrow{\\$} \text{Prove}(\text{CRS}, \mathbf{x}, \mathbf{w})</math></p> <p>08 <b>return</b> <math>(\text{CRS}, \pi)</math></p>
--	---

**Fig. 3.** Real and Sim experiments for the zero-knowledge property

**Definition 6 (Witness Indistinguishable for OR-Composition).** Let  $\mathcal{L}_\vee = \mathcal{L}_0 \vee \mathcal{L}_1$  be an OR-relation. A NIP is  $\Delta$ -Witness Indistinguishable for  $\mathcal{L}_\vee$ , if for the statement  $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1)$  and the witness  $(\mathbf{w}_0, \mathbf{w}_1)$  such that  $\mathbf{x}_0 \in_{\mathbf{w}_0} \mathcal{L}_0 \vee \mathbf{x}_1 \in_{\mathbf{w}_1} \mathcal{L}_1$ , the statistical distance between the output distributions of the **Game 0** and the **Game 1** as defined in Fig. 4 is at most  $\Delta$ .

<p><b>Game 0:</b></p> <p>01 <math>\text{CRS} \xleftarrow{\\$} \text{Setup}(1^\lambda)</math></p> <p>02 <math>\pi \xleftarrow{\\$} \text{Prove}(\text{CRS}, \mathbf{x}, \mathbf{w}_0)</math></p> <p>03 <b>return</b> <math>\pi</math></p>	<p><b>Game 1:</b></p> <p>04 <math>\text{CRS} \xleftarrow{\\$} \text{Setup}(1^\lambda)</math></p> <p>05 <math>\pi \xleftarrow{\\$} \text{Prove}(\text{CRS}, \mathbf{x}, \mathbf{w}_1)</math></p> <p>06 <b>return</b> <math>\pi</math></p>
--	--

**Fig. 4.** Real and Sim experiments for the witness-indistinguishability

Moreover, if  $\Delta = 0$ , NIP is perfectly witness indistinguishable.

We define NIZK as NIP that satisfy completeness, adaptive soundness and zero-knowledge property while for NIWI, the zero-knowledge property is replaced with witness-indistinguishability.

### 3 From Interactive to Non-interactive

One of the most common way to construct a non-interactive zero-knowledge proof is via the Fiat-Shamir [23] transform. However, we additionally require the zero-knowledge property to be ROM-free, which is not the case using this transform. The two existing variants available for  $\Sigma$ -protocols (1-round protocols) are [29] and its more efficient and more generic improvement [15].

**Lindell’s Transform.** [29] In Lindell’s transform, the challenge of  $\Sigma$ -protocol is of the form  $H(\mathbf{x}, \text{Com}(\mathbf{R}))$ , where  $\mathbf{R}$  is the first round message of the  $\Sigma$ -protocol and  $\text{Com}$  is a dual-mode commitment [29] (aka. hybrid trapdoor commitment [11]). However, if we want to generalize this transform to multi-round PCIP, this approach is not very efficient. That is because, we need to include the commitments and the decommitments of every round of PCIP into the final proof. Moreover, following the generic construction of dual-mode commitment from PCIP schemes in [29], the size of one commitment and one decommitment is equal to the size of one PCIP proof. Therefore, if we directly

apply the Lindell’s transform, we will have a proof size blow-up of factor  $O(n)$ , where  $n$  is the number of rounds. Consequently, it may lose the efficiency gain of multi-round PCIP schemes over  $\Sigma$ -protocols.

**Ciampi et al. Transform [15]** The transform in [15] requires only computational optimal soundness (weaker than special soundness) and computational HVZK of the underlying interactive protocols, and it is more efficient than [29]. However, the [15] transform relies heavily on the existence of an OR-composition of interactive protocols. Unfortunately, the most efficient interactive lattice-based proof systems are all 2-round protocols [4, 5, 21], and the previous OR-compositions of interactive proof systems [1, 16, 24] cannot be applied to multi-round PCIPs.

In this section, we further improve the [15] transform by extending it to support OR-composition of an  $n_0$ -round computational HVZK and round-by-round sound PCIP<sub>0</sub> and an  $n_1$ -round computational HVZK and round-by-round sound PCIP<sub>1</sub>. Notice that if we apply our transform to two 1-round PCIPs ( $\Sigma$ -protocols), the resulting NIZK scheme is almost as efficient as in [15]. More precisely, in the case of  $\Sigma$ -protocol, we only have two more elements  $(a_0, a_1) \in \mathbb{Z}_{\ell_{1,0}} \times \mathbb{Z}_{\ell_{1,1}}$  than [15], where  $\ell_{1,0}, \ell_{1,1}$  are the size of the challenge spaces of PCIP<sub>0</sub> and PCIP<sub>1</sub>. In Sect. 2.3 we recall the definitions of two different types of non-interactive proofs NIP: NIZK proofs and Non-Interactive Witness Indistinguishable (NIWI) proofs.

### 3.1 Construction of Our OR-Proof

We recall that the intuition behind our OR-proof is explained in Sect. 1.2. We then directly give the construction of our OR-proof in this section.

Let PCIP<sub>0</sub> (resp. PCIP<sub>1</sub>) be an  $n_0$ -round (resp.  $n_1$ -round) public coin interactive proof for proving the membership of two languages  $\mathcal{L}_0$  and  $\mathcal{L}_1$ , and we denote the size of challenge spaces by  $(\ell_{1,0}, \dots, \ell_{n_0,0}, \ell_{1,1}, \dots, \ell_{n_1,1})$ . The goal is to prove that  $\mathbf{x}_0 \in \mathcal{L}_0$  or  $\mathbf{x}_1 \in \mathcal{L}_1$  without revealing exactly which witness is used. The idea behind this proof, using  $\mathbf{w}_b$ , is to first sample a random offset  $\mathbf{A}_b = (a_{1,b}, \dots, a_{n_b,b})$ . Then, we simulate the proof PCIP<sub>1-b</sub> for which we don’t have a witness to build the second offset  $(a_{1,1-b}, \dots, a_{n_{1-b},1-b})$ , which depends on  $\mathbf{A}_b$  and on the commitments  $\{\mathbf{R}_{i,1-b}\}_{j=1}^{n_{1-b}}$ . Finally, we can use  $\mathbf{A}_{1-b}$  to build the proof PCIP<sub>b</sub> for which we know the witness. To verify the proof, we first verify that all the  $\{h_{i,b}\}$  have been correctly generated, then that both proofs pass their verification algorithm.

We give our transform in pseudo-code in Fig. 5. We define  $\mathcal{C}_{i,b}$  as the challenge space of  $i$ -th round of PCIP<sub>b</sub>, we assume that  $\mathcal{C}_{i,b}$  is isomorphic to the additive group  $(\mathbb{Z}_{\ell_{i,b}}, +)$ .

**Properties of Our NIP.** We will prove in the remaining part of this section that the non-interactive proof NIP constructed as in Fig. 5 is correct (Theorem 1), witness-indistinguishable (Theorem 2), and adaptively sound (Theorem 3), if the underlying protocols PCIP<sub>0</sub>, PCIP<sub>1</sub> are both correct, HVZK and round-by-round sound. Moreover, if PCIP<sub>0</sub> and PCIP<sub>1</sub> are both perfectly HVZK, then the resulting NIP is a NIWI proof with perfect witness-indistinguishability.

```

Prove( $\mathbf{x}_0, \mathbf{x}_1, \mathbf{w}_b$ ):
01  $\mathbf{A}_b := (a_{1,b}, \dots, a_{n_b,b}) \xleftarrow{\$} \mathbb{Z}_{\ell_{1,b}} \times \dots \times \mathbb{Z}_{\ell_{n_b,b}}$ 
02  $\text{Trans}_{1-b} \xleftarrow{\$} \text{PCIP}_{1-b}.\text{Sim}(1^\lambda, \mathbf{x}_{1-b})$ 
03  $(\{\mathbf{R}_{i,1-b}, \mathbf{h}_{i,1-b}\}_{i=1}^{n_{1-b}}, \mathbf{s}_{1-b}) := \text{Trans}_{1-b}$ 
04 for  $i = 1..n_{1-b}$  do
05    $a_{i,1-b} \leftarrow \mathbf{h}_{i,1-b} - \mathbf{H}(\{\mathbf{R}_{j,1-b}\}_{j=1}^i, \mathbf{A}_b)$ 
06  $\mathbf{A}_{1-b} \leftarrow (a_{1,1-b}, \dots, a_{n_{1-b},1-b})$ 
07  $\mathbf{st}_{0,b} = \emptyset; \mathbf{h}_{0,b} = \perp$ 
08 for  $i = 1..n_b$  do
09    $(\mathbf{R}_{i,b}, \mathbf{st}_{i,b}) \xleftarrow{\$} \text{PCIP}_b.\text{Prove}_i(\mathbf{st}_{i-1,b}, \mathbf{h}_{i-1,b}, \mathbf{x}_b, \mathbf{w}_b)$ 
10    $\mathbf{h}_{i,b} := \mathbf{H}(\{\mathbf{R}_{j,b}\}_{j=1}^i, \mathbf{A}_{1-b}) + a_{i,b}$ 
11  $\mathbf{s}_b \xleftarrow{\$} \text{PCIP}_b.\text{Prove}_{n_b}(\mathbf{st}_{n_b-1,b}, \mathbf{h}_{n_b-1,b}, \mathbf{x}_b, \mathbf{w}_b)$ 
12 return  $\pi := (\{\mathbf{R}_{i,0}\}_{i=1}^{n_0}, \{\mathbf{R}_{i,1}\}_{i=1}^{n_1}, \mathbf{A}_0, \mathbf{A}_1, \mathbf{s}_0, \mathbf{s}_1)$ 
Verif( $\mathbf{x}_0, \mathbf{x}_1, \pi$ ):
13 for  $i = 1..n_0$  do
14    $\mathbf{h}_{i,0} := \mathbf{H}(\{\mathbf{R}_{j,0}\}_{j=1}^i, \mathbf{A}_1) + a_{i,0}$ 
15 for  $i = 1..n_1$  do
16    $\mathbf{h}_{i,1} := \mathbf{H}(\{\mathbf{R}_{j,1}\}_{j=1}^i, \mathbf{A}_0) + a_{i,1}$ 
17  $\text{Trans}_0 := (\{\mathbf{R}_{i,0}, \mathbf{h}_{i,0}\}_{i=1}^{n_0}, \mathbf{s}_0)$ 
18  $\text{Trans}_1 := (\{\mathbf{R}_{i,1}, \mathbf{h}_{i,1}\}_{i=1}^{n_1}, \mathbf{s}_1)$ 
19 if  $\text{PCIP}_0.\text{Verify}(\mathbf{x}_0, \text{Trans}_0) = 1 \wedge \text{PCIP}_1.\text{Verify}(\mathbf{x}_1, \text{Trans}_1) = 1$  then
20   return 1
21 else return 0

```

**Fig. 5.** In this figure, we construct an NIP system  $\Pi = (\text{Setup}, \text{Prove}, \text{Verif})$ , which is an OR-composition to prove that  $\mathbf{x}_0 \in \mathcal{L}_0 \vee \mathbf{x}_1 \in \mathcal{L}_1$ . We recall that all challenge spaces are considered as an additive group. Namely, for all operations in the  $i$ -th round of  $\text{PCIP}_b$  are modulo  $\mathbb{Z}_{\ell_{i,b}}$ .

**Theorem 1 (Correctness).** *If  $\text{PCIP}_0$  and  $\text{PCIP}_1$  are both  $\rho$ -correct and  $\Delta$ -HVZK, then  $\Pi$  is  $2\rho + \Delta$ -correct.*

*Proof.* We can observe that in the resulting proof  $\pi$ , we have randomly chosen a bit  $b$ , and the proof  $\pi$  can be divided into two parts  $(\pi_0, \pi_1)$ , where  $\pi_b = (\{\mathbf{R}_{i,b}\}_{i=1}^{n_b}, \mathbf{A}_b, \mathbf{s}_b)$  is an honestly generated proof of  $\text{PCIP}_b$  with correctness error at most  $\rho$ , and  $\pi_{1-b}$  is a simulated transcript of  $\text{PCIP}_{1-b}$  with correctness error at most  $\rho + \Delta$ . Therefore, by the union bound over the correctness of  $\pi_0$  and  $\pi_1$ , we have  $\pi$  has correctness error at most  $2\rho + \Delta$ .  $\square$

**Theorem 2 (Witness-Indistinguishability).** *If  $\text{PCIP}_0$  and  $\text{PCIP}_1$  are two  $\Delta$ -HVZK  $(n_0, n_1)$ -rounds public-coin interactive proofs for the language  $\mathcal{L}_0$  and  $\mathcal{L}_1$  respectively, then  $\Pi$  is  $2\Delta$ -Witness-Indistinguishable. Namely, given a statement  $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1)$  such that  $\mathbf{x}_0 \in_{\mathbf{w}_0} \mathcal{L}_0 \wedge \mathbf{x}_1 \in_{\mathbf{w}_1} \mathcal{L}_1$ , the statistical distance between the proof generated using  $\mathbf{w}_0$  and the one generated using  $\mathbf{w}_1$  is at most  $2\Delta$ .*

**Theorem 3 (Adaptive Soundness).** For  $b \in \{0, 1\}$ , let  $\text{PCIP}_b$  be an  $n_b$ -round round-by-round  $\varepsilon'$ -sound interactive protocol, then  $\Pi$  is  $(t, \varepsilon, Q_H)$ -adaptively sound, where

$$t = \text{poly}(\lambda), \quad \varepsilon \leq (Q_H + 2n)^2 \cdot n \cdot \varepsilon',$$

with  $n = \max(n_0, n_1)$ .

*Proof.* Assuming  $\mathcal{A}$  a PPT adversary, running in polynomial time  $t$ , wins the adaptive soundness game within probability  $\varepsilon$  by generating a valid OR-proof  $\pi$  for  $(\mathbf{x}_0, \mathbf{x}_1)$  where  $\mathbf{x}_0 \notin \mathcal{L}_0$  and  $\mathbf{x}_1 \notin \mathcal{L}_1$ ,

$$\pi = (\{R_{i,0}\}_{i=1}^{n_0}, \{R_{i,1}\}_{i=1}^{n_1}, A_0, A_1, s_0, s_1).$$

Moreover, we can compute  $A_0 = (a_{1,0}, \dots, a_{n_0,0})$ ,  $A_1 = (a_{1,1}, \dots, a_{n_1,1})$ , and  $h_{i,b} = H(\{R_{j,b}\}_{j=1}^i, A_{1-b}) + a_{i,b}$ . We give the security proof via a sequence of games:

- **Game<sub>0</sub>** : The **Game<sub>0</sub>** is the original adaptive soundness game.
- **Game<sub>1</sub>** : In this game, we assume that for  $i_0 \in [n_0], i_1 \in [n_1]$ , all the queries of the form  $(\{R_{i,0}\}_{j=1}^{i_0}, A_1)$  and  $(\{R_{i,1}\}_{j=1}^{i_1}, A_0)$  have been queried to the random oracle. Remind that if the adversary  $\mathcal{A}$  does not fulfil this condition, we can construct a new adversary  $\mathcal{B}$  that additionally makes the above two queries with the same running time and winning probability against the adaptive soundness game. Therefore, we have  $\text{Adv}_0 = \text{Adv}_1$ , but the number of queries has slightly increased  $Q'_H = Q_H + n_0 + n_1$ .

*Analysis of the Winning Probability  $\text{Adv}_1$  of  $\mathcal{A}$  in **Game<sub>1</sub>**.* We define the bit  $b \in \{0, 1\}$  such that there is a random oracle query of the form  $(\cdot, A_b)$  happens before any query of the form  $(\cdot, A_{1-b})$ .

Since  $\pi$  is a valid proof, we have for  $i \in [n_b]$  that  $h_{i,b} = H(\{R_{j,b}\}_{j=1}^i, A_{1-b}) + a_{i,b}$ . Note that in the proof given by the adversary is of the form  $\pi = (\pi_0, \pi_1)$  where any query of the form  $(\cdot, A_{1-b})$  happens after a query of the form  $(\cdot, A_b)$ . Therefore, the adversary  $\mathcal{A}$  can only choose at most  $Q_H + 2n$  different offsets as  $A_b$ . Moreover, for all  $i \in [n_b]$ , given  $\{R_{j,b}\}_{j=1}^i$  and  $A_b = (\{a_{j,b}\}_{j=1}^{n_b})$ , there are at most  $Q_H + 2n$  different challenge values  $h_{i,b} := H(\{R_{j,b}\}_{j=1}^i, A_{1-b}) + a_{i,b}$  depending on the choice of  $A_{1-b}$ . Thus, the adversary has in total at most  $(Q_H + 2n)^2$  choices of  $h_{i,b}$ .

We emphasize that the output distribution of the random oracle is uniformly random. Therefore, the distribution of  $h_{i,b}$  conditioned on the choice of  $A_b, A_{1-b}$  is still uniformly random by using the One-Time Pad argument.

We recall that, for the round-by-round  $\varepsilon$ -soundness, for all  $j \in [n_b]$ , given the prover's messages  $(\{R_{i,0}\}_{i=1}^j)$ , if the challenge is selected uniformly, the partial transcript has probability  $1 - \varepsilon$  to be "doomed". The adversary has  $(Q_H + 2n)^2$  choices over  $(A_b, A_{1-b})$ . On the other hand, the total transcript is in the "doomed set" with probability  $1 - (1 - \varepsilon')^n \leq n \cdot \varepsilon'$ . Therefore, we have that the success probability for the adversary in finding a pair of  $(A_b, A_{1-b})$  such that the transcript of the side  $b$  is not doomed is at most  $(Q_H + 2n)^2 \cdot n \cdot \varepsilon'$ .

Summarizing all the hybrid games, we have

$$t = \text{poly}(\lambda), \quad \varepsilon \leq (Q_H + 2n)^2 \cdot n \cdot \varepsilon'.$$

□

### 3.2 Adaptively Sound Non-Interactive Zero-Knowledge Proof

We follow the same framework of [15] for defining a transform from  $n$ -round interactive proof systems to NIZK: we use our OR-composition in Sect. 3.1 to let the prover combine the interactive proof system with a proof of hard membership problem.

Since the transform of [15] (and ours) makes use of a membership-hard language  $\mathcal{L}$ , let us first define it in Definition 7.

**Definition 7 (NP membership problem[29]).** A language  $\mathcal{L}$  is a  $(t, \varepsilon_{\mathcal{L}})$ -hard NP membership language if there exists a PPT sampler  $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$  such that for every PPT distinguisher  $\mathcal{D}$ , running in polynomial time  $t$ , we have

$$|\Pr[\mathcal{D}(\mathcal{S}_0(1^\lambda), 1^\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{S}_1(1^\lambda), 1^\lambda) = 1]| \leq \varepsilon_{\mathcal{L}},$$

where  $\mathcal{S}$  behaves as follows

- $\mathcal{S}_0(1^\lambda)$  samples  $(\mathbf{x}_0, \mathbf{w}_0) \stackrel{\$}{\leftarrow} \mathcal{L}$ , and returns  $\mathbf{x}_0$ .
- $\mathcal{S}_1(1^\lambda)$  samples  $\mathbf{x}_1 \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda \setminus \mathcal{L}$ , and returns  $\mathbf{x}_1$ .

**Transform from Interactive to Non-Interactive.** Given a language  $\mathcal{L}_0$  and an instance  $\mathbf{x}_0 \in_{\mathbf{w}_0} \mathcal{L}_0$ , our goal is to prove that  $\mathbf{x}_0 \in \mathcal{L}_0$  without leaking any additional information about  $\mathbf{w}_0$ . We follow the same overall framework as [15] by adding a membership-hard language  $\mathcal{L}_1$  together with an instance  $\mathbf{x}_1 \in \mathcal{L}_1$ , then the NIZK proof consists of a proof that  $\mathbf{x}_0 \in \mathcal{L}_0 \vee \mathbf{x}_1 \in \mathcal{L}_1$ , and  $(\mathbf{x}_1, \mathcal{L}_1)$  is the CRS of the NIZK proof system. We give below some intuitions behind the soundness and the zero-knowledge property of this general construction.

- **Soundness:** As  $\mathcal{L}_1$  is a membership-hard problem, we can switch  $\mathbf{x}_1 \in_{\mathbf{w}_1} \mathcal{L}_1$  into  $\mathbf{x}'_1 \in \{0, 1\}^\lambda \setminus \mathcal{L}_1$  without the adversary noticing it. Since  $\mathbf{x}'_1 \in \{0, 1\}^\lambda \setminus \mathcal{L}_1$ , a valid proof  $\pi$  for the fact that  $\mathbf{x}_0 \in \mathcal{L}_0 \vee \mathbf{x}'_1 \in \mathcal{L}_1$  directly implies that  $\mathbf{x}_0 \in \mathcal{L}_0$ .
- **Zero-Knowledge:** We can simulate every proof using  $\mathbf{w}_1$  instead of  $\mathbf{w}_0$ . By the witness-indistinguishability of the OR-proof, this change is oblivious for the adversary. This proves the zero-knowledge property of the NIZK proof system.

Formally, let  $\text{PCIP}_0$  be a  $(k, \ell)$ -sound  $n_0$ -round interactive proof system for the NP language  $\mathcal{L}_0$ . We will consider a  $(t, \varepsilon_{\mathcal{L}})$ -hard NP membership  $\mathcal{L}_1$  and its associated interactive proof system  $\text{PCIP}_1$ . Let  $\Pi$  denote the NIWI scheme obtained by applying the OR-composition from Sect. 3.1 to  $\text{PCIP}_0$  and  $\text{PCIP}_1$ . We give the explicit transform from an IP protocol  $\text{PCIP}_0$  to a NIZK scheme  $\Sigma$  in Fig. 6.

The correctness of  $\Sigma$  is straightforward from Theorem 1:

**Theorem 4 (Correctness).** If  $\text{PCIP}_0$  and  $\text{PCIP}_1$  are both at least  $\rho$ -correct and  $\Delta$ -HVZK, then  $\Sigma$  is  $2\rho + \Delta$ -correct.

**Theorem 5 (Zero-Knowledge).** If  $\text{PCIP}_0$  and  $\text{PCIP}_1$  are both  $\Delta$ -HVZK multi-round  $(n_0, n_1$  rounds respectively) interactive protocols, then  $\Sigma$  is  $2\Delta$ -Zero-Knowledge.

*Proof.* Since we have  $\mathbf{x}_1 \in_{\mathbf{w}_1} \mathcal{L}_1$ , we can use  $\mathbf{w}_1$  to compute the NIWI proof, which simulates an honestly generated proof with statistical distance at most  $2\Delta$  by Theorem 2. □

<b>Setup</b> ( $1^\lambda$ ): 01 $(\mathbf{x}_1, \mathbf{w}_1) \xleftarrow{\$} \mathcal{L}_1$ 02 $\text{CRS} := \mathbf{x}_1$ 03 <b>return</b> CRS	<b>Prove</b> (CRS, $\mathbf{x}_0, \mathbf{w}_0$ ): 04 $\pi \xleftarrow{\$} \Pi.\text{Prove}((\mathbf{x}_0, \text{CRS}), \mathbf{w}_0)$ 05 <b>return</b> $\pi$  <b>Verif</b> (CRS, $\mathbf{x}, \pi$ ): 06 <b>return</b> $\Pi.\text{Verif}((\mathbf{x}, \text{CRS}), \pi)$
---	--

**Fig. 6.** Transform from an optimal-sound interactive protocol  $\text{PCIP}_0$  into adaptively sound NIZK scheme  $\Sigma$ .

**Theorem 6 (Adaptive Soundness).** For  $b \in \{0, 1\}$ , let  $\text{PCIP}_b$  be a  $\varepsilon_b$ -Round-By-Round sound  $n_b$ -round interactive protocol such that  $\text{PCIP}_1$  is the interactive proof associated to a  $(t', \varepsilon'_\mathcal{L})$ -hard NP membership language  $\mathcal{L}_1$ , then  $\Sigma$  is  $(t, \varepsilon, \mathbf{Q}_H)$ -adaptively sound, where

$$t \approx t', \quad \varepsilon \leq (\mathbf{Q}_H + 2n)^2 \cdot n \cdot \varepsilon' + \varepsilon'_\mathcal{L},$$

with  $\varepsilon' = \max(\varepsilon_0, \varepsilon_1)$  and  $n = \max(n_0, n_1)$ .

*Proof.* We will give a simple game-based proof of this theorem. There are only 2 hybrids described as in Fig. 7.

<b>Exp</b> <sub>AdSnd</sub> ( $1^\lambda$ ): 01 $(\mathbf{x}_1, \mathbf{w}_1) \xleftarrow{\$} \mathcal{L}_1$ 02 $\mathbf{x}_1 \xleftarrow{\$} \{0, 1\}^\lambda \setminus \mathcal{L}_1$ 03 $\text{CRS} := \mathbf{x}_1$ 04 $(\mathbf{x}^*, \pi^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\text{Hash}}(\text{CRS})}$ 05 <b>if</b> $\mathbf{x}^* \in \{0, 1\}^n \setminus \mathcal{L}_0 \wedge \text{Verif}(\text{CRS}, \mathbf{x}^*, \pi^*) = 1$ <b>then</b> 06 <b>return</b> 1 07 <b>else return</b> 0	 // <b>Game</b> <sub>0</sub> // <b>Game</b> <sub>1</sub>	<b>O</b> <sub>Hash</sub> ( $\mathbf{R}$ ): 08 $\mathbf{h} \xleftarrow{\$} \mathcal{C}$ 09 <b>return</b> $\mathbf{h}$
--	--	--

**Fig. 7.** The security games for proving the adaptive soundness of  $\Sigma$ . The line commented with **Game** <sub>$i$</sub>  is the pseudo-code that only exists in  $i$ -th hybrid.

The **Game**<sub>0</sub> is the original security game for the adaptive soundness of  $\Sigma$ . In game **Game**<sub>1</sub>, the only difference is that  $\mathbf{x}_1$  in CRS is chosen from the set  $\{0, 1\}^\lambda \setminus \mathcal{L}_1$ . Therefore, we have

$$\text{Adv}_0 = \varepsilon, \quad |\text{Adv}_1 - \text{Adv}_0| \leq \varepsilon'_\mathcal{L}.$$

where  $\text{Adv}_0$  (respectively  $\text{Adv}_1$ ) is the advantage of  $\mathcal{A}$  in game **Game**<sub>0</sub> (respectively **Game**<sub>1</sub>). Moreover, in **Game**<sub>1</sub>, since  $\mathbf{x}_1$  is not in  $\mathcal{L}_1$  and  $\mathbf{x}_0$  is neither in  $\mathcal{L}_0$ ,  $\pi$  is a valid attack for the underlying NIWI scheme. Therefore, we have  $\text{Adv}_1 \leq (\mathbf{Q}_H + 2n)^2 \cdot n \cdot \varepsilon'$  from Theorem 3. Combining hybrids together we have  $t \approx t'$  and

$$\varepsilon \leq (\mathbf{Q}_H + 2n)^2 \cdot n \cdot \varepsilon' + \varepsilon'_\mathcal{L}.$$

□



## 4 Security of Our Transform in the Quantum Random Oracle Model

In this section, we give a security proof of our OR-composition from two public-coin interactive proofs ( $n_0$ -round and  $n_1$ -round respectively) into one NIZK in the quantum random oracle model. Note that we can straightforwardly extend our proof in the QROM to our transform from PCIP to NIZK as described in Sect. 3.2.

While it is an important achievement to prove security in the QROM for post-quantum primitives, there is a price that one has to pay. One drawback is that there is a significant loss in the security argument. The second one is related to the programmability of the random oracle: proofs that were in the NPRM in the classical setting now need the quantum random oracle to be programmable in the security reduction. The last one is that we cannot prove our transform for round-by-round sound PCIP with acceptable security loss (polynomial in the number of rounds), due to the fact that we need to reprogram every round to fulfill a reduction, which introduces an exponential security loss in the number of rounds. Therefore, we limit our transform to optimal-sound PCIP protocols. Firstly, we introduce the notion of answerable challenge and provide the formal definition of optimal-soundness.

**Definition 8 (Answerable Challenges).** Let  $\text{Ans}(\text{Trans}_i, h_i)$  be a function that takes a partial transcript until  $i$ -th round  $\text{Trans}_i = (\{R_j, h_j\}_{j=1}^{i-1}, R_i)$  and a challenge  $h_i$  as input, and returns 1 if there exists  $\text{Trans}' = (\{R_j, h_j\}_{j=i+1}^n, s)$  such that  $(\text{Trans}_i, h_i, \text{Trans}')$  is a valid transcript and 0 otherwise. We say that a challenge  $h_i$  is an **answerable challenge** for round  $i$  if  $\text{Ans}(\text{Trans}_i, h_i) = 1$ .

We emphasize that the function  $\text{Ans}$  can be a non-efficiently computable function here.

**Definition 9 (Optimal Soundness).** Let  $\mathcal{L}$  be an NP language, we say that PCIP is  $(k, \ell, i)$ -optimal sound if, for all statement not in the language  $x \notin \mathcal{L}$ , and for all partial transcripts  $\text{Trans}_i = (\{R_j, h_j\}_{j=1}^{i-1}, R_i)$  there exist at most  $k$  answerable challenges  $\{h_i^{(j)}\}_{j \in [k]}$  such that  $\text{Ans}(\text{Trans}_i, h_i^{(j)}) = 1$  for all  $j \in [k]$  and the size of the  $i$ -th challenge space is at least  $2^\ell$ .

We note that, the optimal soundness is implied by the special soundness which is the case for most PCIP protocols. Moreover optimal soundness straightforwardly imply the negligible soundness, while the latter one is equivalent to the round-by-round soundness in our case. Thus, limiting our transform to the PCIPs with optimal soundness is indeed a restriction.

We will use the measure-and-reprogram 2.0 technique proposed in [19] and we apply it to our NIZK transform in the same way that [19] apply it for proving sequential-OR proof. Firstly, we give a quick overview of the measure-and-reprogram 2.0 technique proposed in [19].

**Measure-and-Reprogram 2.0, Multiple Input** [19]. Let  $\mathcal{A}$  be a quantum adversary that has  $Q_H$  quantum queries to a random oracle  $H : \mathcal{X} \rightarrow \mathcal{Y}$ , where  $\mathcal{X}, \mathcal{Y}$  are both finite non-empty sets. Assuming that for a predicate (possibly quantum and not efficiently computable)  $\Gamma$ , the adversary  $\mathcal{A}$  can output in polynomial time  $t$  a transcript  $\text{Trans} = (X_0, \dots, X_{n-1}, z)$  such that  $\Gamma(\text{Trans}, H(X_0), \dots, H(X_{n-1})) = \text{True}$ .

The goal is to build a multi-stage simulator  $\mathcal{R}^{\mathcal{A}}$  such that stage by stage it outputs  $X_i$ 's and takes the corresponding  $\Theta_i$ 's as input and finally outputs a (possibly quantum)  $z$  such that for the same predicate we have  $\Gamma(X_0, \dots, X_{n-1}, z, \Theta_0, \dots, \Theta_{n-1}) = \text{True}$ .

Don et al. [19] showed the existence of a quantum adversary  $\mathcal{R}^{\mathcal{A}}$  that proceeds as follows: Firstly, it outputs a permutation  $\sigma$  together with a hash input  $\mathbf{x}_{\sigma(0)}$  and it takes as input  $\Theta_{\sigma(0)}$  from a third party  $\mathcal{V}$ . Then for every stage  $0 < i \leq n - 1$ ,  $\mathcal{R}^{\mathcal{A}}$  outputs a hash input  $\mathbf{x}_{\sigma(i)}$  and it takes as input  $\Theta_{\sigma(i)}$  from  $\mathcal{V}$ . Finally, it outputs a possibly quantum  $z$ . We denote this procedure as  $(\sigma, \sigma(X), z) \stackrel{\$}{\leftarrow} \langle \mathcal{R}^{\mathcal{A}}, \sigma(\Theta) \rangle$ , where  $X = (X_0, \dots, X_{n-1})$  and  $\Theta = (\Theta_0, \dots, \Theta_{n-1})$ . In the special case of PCIP protocols,  $\mathcal{V}$  refers to the verifier.

More precisely, we have the following theorem:

**Theorem 7** ([19, Theorem 6]). *Let  $\mathcal{X}$  and  $\mathcal{Y}$  be the input and output sets of the hash function  $H : \mathcal{X} \rightarrow \mathcal{Y}$ . Let  $\mathcal{A}$  be a polynomial time oracle quantum algorithm that makes  $Q_H$  random oracle queries to  $H$  and outputs an  $n$ -dimensional vector  $X = (X_0, \dots, X_{n-1})$  and a possibly quantum  $z$ . There exists a  $(n + 1)$ -stage quantum algorithm  $\mathcal{R}^{\mathcal{A}}$  that behaves as described above, satisfying the following property: For any  $X^* \in \mathcal{X}^n$  without duplicate entries and for any predicate (possibly quantum and not efficiently computable)  $\Gamma$ , and a third party  $\mathcal{V}$ , we have:*

$$\Pr \left[ \begin{array}{l} X = X^* \wedge \\ \Gamma(X, \Theta, z) \end{array} \mid (\sigma, \sigma(X), z) \stackrel{\$}{\leftarrow} \langle \mathcal{R}^{\mathcal{A}}, \sigma(\Theta) \rangle \right] \geq \frac{1}{(2Q_H + 1)^{2n}} \cdot \Pr \left[ \begin{array}{l} X = X^* \wedge \\ \Gamma(X, H(X), z) \end{array} \mid (X, z) \stackrel{\$}{\leftarrow} \mathcal{A}(1^\lambda) \right] \quad (3)$$

**Application to Our Zero-Knowledge Proof.** Formally, given a  $n_0$ -round  $\text{PCIP}_0$  and a  $n_1$ -round  $\text{PCIP}_1$ , for languages  $\mathcal{L}_0$  and  $\mathcal{L}_1$  respectively. We proposed a non-interactive proof of the form  $\pi_{\mathcal{V}} = (A_0, A_1, \{R_{i,0}\}_{i=0}^{n_0}, \{R_{i,1}\}_{i=1}^{n_1}, s_0, s_1)$  for the language  $\mathcal{L}_{\mathcal{V}} = \{(x_0, x_1) : x_0 \in \mathcal{L}_0 \vee x_1 \in \mathcal{L}_1\}$ .

We assume that for  $b \in \{0, 1\}$ , the interactive protocol  $\text{PCIP}_b$  is  $(k_b, \ell_b, i_b)$ -optimal sound, and we have  $i_b^*$  such that given the first  $i_b^*$  elements  $R_{1,b}, \dots, R_{i_b^*,b}$ , there are only  $k_b$  answerable challenges. This property is captured by the answerable predicates given in the optimal soundness  $\text{Ans}_b(R_{1,b}, \dots, R_{i_b^*,b}, h_{i_b^*,b})$ .

**Theorem 8.** *For  $b \in \{0, 1\}$ , let  $\text{PCIP}_b$  be a  $(k, \ell, i_b)$ -optimal sound  $n_b$ -round interactive protocol. Let  $\Pi_{\mathcal{V}}$  be the non-interactive zero-knowledge proof given by applying our transform in Sect. 3.1. Any quantum adversary  $\mathcal{A}$  running in time  $t$ , making  $Q_H$  quantum random oracle, breaks the adaptive soundness of  $\Pi_{\mathcal{V}}$  with probability at most  $\frac{k}{2^{\ell}} \cdot (2Q_H + 1)^4$ .*

*Proof.* Assuming  $\mathcal{A}$  is a quantum adversary making  $Q_H$  quantum random oracle queries against the adaptive soundness of  $\Pi_V$ . By the definition of adaptive soundness, given two false statements  $\mathbf{x}_0 \notin \mathcal{L}_0$  and  $\mathbf{x}_1 \notin \mathcal{L}_1$ ,  $\mathcal{A}$  can generate a valid proof  $\pi_V = (A_0, A_1, \{R_{i,0}\}_{i=1}^{n_0}, \{R_{i,1}\}_{i=1}^{n_1}, s_0, s_1)$  with non-negligible advantage. For simplicity, we denote the challenge by,

$$h_{i,b} := H(\{R_{j,b}\}_{j=1}^i, A_{1-b}) + a_{i,b}. \tag{4}$$

Note that, in our non-interactive proof construction  $h_{i,b}$  is used as the challenges in the underlying interactive protocols. Since  $\pi_V$  is a valid proof, for  $b \in \{0, 1\}$ , and  $i \in [n_b]$ , we have  $\text{Ans}_b(\{R_{j,b}\}_{j=1}^i, h_{j,b}) = \text{True}$ .

For our convenience, we will consider an adversary  $\mathcal{A}'$  that proceeds exactly like  $\mathcal{A}$ , except that it only outputs a partial proof  $\pi' = (\{R_{i,0}\}_{i=1}^{i_0^*}, A_1, \{R_{i,1}\}_{i=1}^{i_1^*}, A_0)$ . For more compact notation, we denote  $X_b = (\{R_{i,b}\}_{i=1}^{i_b^*}, A_{1-b})$  for  $b \in \{0, 1\}$ . We also define a predicate  $\Gamma$  as follows:

$$\Gamma((X_0, X_1), (H(X_0), H(X_1))) = \text{Ans}_0(\{R_{i,0}\}_{i=1}^{i_0^*}, h_{i_0^*,0}) \wedge \text{Ans}_1(\{R_{i,1}\}_{i=1}^{i_1^*}, h_{i_1^*,1}).$$

Here, we recall that  $h_{i_b^*,b}$  can be computed by using  $\pi', H(X_0), H(X_1)$  as in Equation (4). By the definition of the answerable challenge predicate, assuming a valid proof  $\pi_V$ , the corresponding partial proof  $\pi' = (X_0, X_1)$  verifies that  $\Gamma((X_0, X_1), (H(X_0), H(X_1))) = \text{True}$ .

Now, it is easy to see that  $(\mathcal{A}', \Gamma)$  fits into the requirement of Theorem 7. By simply applying Theorem 7, for all  $(X_0^*, X_1^*)$ , two uniformly chosen  $\Theta_0, \Theta_1$  and two instances  $(\mathbf{x}_0, \mathbf{x}_1)$ , we have an adversary  $\mathcal{B}$  such that:

$$\begin{aligned} & \Pr \left[ \begin{array}{l} X_0 = X_0^* \wedge \\ X_1 = X_1^* \wedge \\ \Gamma((X_0, X_1), (\Theta_0, \Theta_1)) \end{array} \middle| (\sigma, \sigma(X_0, X_1), \perp) \stackrel{\$}{\leftarrow} \langle \mathcal{B}(\mathbf{x}_0, \mathbf{x}_1), \sigma(\Theta_0, \Theta_1) \rangle \right] \\ & \geq \frac{1}{(2Q_H + 1)^4} \cdot \Pr \left[ \begin{array}{l} X_0 = X_0^* \wedge \\ X_1 = X_1^* \wedge \\ \Gamma((X_0, X_1), (H(X_0), H(X_1))) \end{array} \middle| \begin{array}{l} (X_0, X_1, \perp) \\ \stackrel{\$}{\leftarrow} \mathcal{A}^H(\mathbf{x}_0, \mathbf{x}_1) \end{array} \right]. \end{aligned} \tag{5}$$

In the final step, we will construct an adversary  $\mathcal{C}$  that helps us to choose  $(\Theta_0, \Theta_1)$ . More precisely, we describe the behavior of  $\mathcal{C}$  as in Fig. 8.

Note that the left side of Equation (5) can be bounded by  $\frac{k}{2^\ell}$ . More precisely, since we have  $\Gamma((X_0, X_1), (\Theta_0, \Theta_1)) = \text{True}$ , we have also  $\text{Ans}_b(X_b, h_b) = \text{True}$ . But, the challenge  $h_b$  is chosen uniformly random by an honest verifier  $\text{Verifier}_b$  in line 09 Fig. 8. Therefore  $\Pr[\text{Ans}_b(X_b, h_b)] \leq \frac{k}{2^\ell}$  by the optimal soundness. Combining this upper bound with Eq. 5, we have the probability of  $\mathcal{A}$  breaking the adaptive soundness of  $\Pi_V$  is at most  $\frac{k}{2^\ell} \cdot (2Q_H + 1)^4$ .  $\square$

$\mathcal{C}(x \notin \mathcal{L})$ : 01 $(\sigma, X_{\sigma(0)}, \text{st}) \stackrel{\$}{\leftarrow} \mathcal{B}_1(x_0, x_1)$ 02 $b \leftarrow \sigma(1)$ 03 $x_b := x; \mathcal{L}_b := \mathcal{L}; \text{Verifier}_b := \text{Verifier}$ 04 $x_{1-b} \stackrel{\$}{\leftarrow} \{0, 1\}^*$ ; $\mathcal{L}_{1-b} \stackrel{\$}{\leftarrow} \{0, 1\}^*$ 05 $\Theta_{\sigma(0)} \stackrel{\$}{\leftarrow} \mathcal{Y}$ 06 $(X_{\sigma(1)}, \text{st}) \stackrel{\$}{\leftarrow} \mathcal{B}_2(\Theta_{\sigma(0)}, \text{st})$ 07 <b>parse</b> $(R_{1,\sigma(1)}, \dots, R_{i_{\sigma(1)},\sigma(1)}, A_{\sigma(1)}) =: X_{\sigma(1)}$ 08 <b>parse</b> $(a_{1,\sigma(1)}, \dots, a_{n_{\sigma(1)},\sigma(1)}) =: A_{\sigma(1)}$ 09 $h_{\sigma(1)} \stackrel{\$}{\leftarrow} \text{Verifier}_{\sigma(1)}(X_{\sigma(1)})$ 10 $\Theta_{\sigma(1)} := h_{\sigma(1)} - a_{i_{\sigma(1)},\sigma(1)}$ 11 $\perp \stackrel{\$}{\leftarrow} \mathcal{B}_3(\Theta_{\sigma(1)}, \text{st})$ 12 <b>return</b> $(X_{\sigma(1)}, h_{\sigma(1)}, \Theta_0, \Theta_1)$
---

**Fig. 8.** Assuming PCIP<sub>0</sub> and PCIP<sub>1</sub> are  $(k, \ell, i)$ -optimal sound, we give the description of the adversary  $\mathcal{C}$  which interacts with the verifier Verifier of the underlying PCIP. Note that  $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$  is a 3-stage algorithm with an internal state st.

**Acknowledgement.** We thank the anonymous reviewers of Asiacrypt 2022 and PKC 2023 for their many insightful suggestions to improve our paper.

This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025). This work is also supported by the PEPR quantique France 2030 programme (ANR-22-PETQ-0008), and by the ANR ASTRID project AMIRAL (ANR-21-ASTR-0016). This work was carried out in the context of Beyond5G, a project funded by the French government as part of the economic recovery plan, namely "France Relance", and the investments for the future program. Adela Georgescu was partly funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER), with the support of Région Bretagne.

## References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer, Heidelberg (2002)
2. 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. editors, ACM CCS 93, pp. 62–73. ACM Press, November (1993)
3. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). [https://doi.org/10.1007/11761679\\_25](https://doi.org/10.1007/11761679_25)
4. Bootle, J., Lyubashevsky, V.: Ngoc Khanh Nguyen, and Gregor Seiler. More efficient amortization of exact zero-knowledge proofs for LWE. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021. Part II, volume 12973 of LNCS, pp. 608–627. Springer, Heidelberg (2021)
5. Bootle, J., Lyubashevsky, V., Seiler, G.: Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Part I, volume 11692 of LNCS, pp. 176–202. Springer, Heidelberg (2019)

6. 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, volume 12172 of LNCS, pp. 738–767. Springer, Heidelberg (2020)
7. 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)
8. Canetti, R., et al.: Fiat-Shamir: from practice to theory. In: Moses Charikar and Edith Cohen, editors, 51st ACM STOC, pp. 1082–1090. ACM Press June (2019)
9. 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. LNCS, vol. 10820, pp. 91–122. Springer, Cham (2018)
10. Canetti, R., Goldreich, O., Halevi, S.: On the random-oracle methodology as applied to length-restricted signature schemes. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 40–57. Springer, Heidelberg (2004)
11. Catalano, D., Visconti, I.: Hybrid commitments and their applications to zero-knowledge proof systems. *Theor. Comput. Sci.* **374**(1–3), 229–260 (2007)
12. Chaum, D.: Blind signature system. In: Chaum, D. (ed.) CRYPTO'83, page 153. Plenum Press, New York, USA (1983)
13. Chen, M.-S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: SOFIA:  $\mathcal{MQ}$ -based signatures in the QROM. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. Part II, volume 10770 of LNCS, pp. 3–33. Springer, Heidelberg (2018)
14. Chen, Y., Lombardi, A., Ma, F., Quach, W.: Does fiat-shamir require a cryptographic hash function? In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 334–363. Springer, Cham (2021)
15. Ciampi, M., Persiano, G., Siniscalchi, L., Visconti, I.: A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A. Part II, volume 9563 of LNCS, pp. 83–111. Springer, Heidelberg (2016)
16. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
17. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
18. De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge proof systems. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 52–72. Springer, Heidelberg (1988)
19. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. Part III, volume 12172 of LNCS, pp. 602–631. Springer, Heidelberg (2020)
20. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Part II, volume 11693 of LNCS, pp. 356–383. Springer, Heidelberg (2019)
21. Esgin, M.F., Nguyen, N.K., Seiler, G.: Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. Part II, volume 12492 of LNCS, pp. 259–288. Springer, Heidelberg (2020). [https://doi.org/10.1007/978-3-030-64834-3\\_9](https://doi.org/10.1007/978-3-030-64834-3_9)
22. Feige, U., Fiat, A., Shamir, A.: Zero knowledge proofs of identity. In: Aho, A., editor, 19th ACM STOC, pp. 210–217. ACM Press, May (1987)
23. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)

24. Fischlin, M., Harasser, P., Janson, C.: Signatures from sequential-OR proofs. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. Part III, volume 12107 of LNCS, pp. 212–244. Springer, Heidelberg (2020)
25. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A., editor, 19th ACM STOC, pp. 218–229. ACM Press, May (1987)
26. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
27. Katsumata, S.: A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12826, pp. 580–610. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84245-1\\_20](https://doi.org/10.1007/978-3-030-84245-1_20)
28. Kim, S., David, J.W.: Multi-theorem preprocessing NIZKs from lattices. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. Part II, volume 10992 of LNCS, pp. 733–765. Springer, Heidelberg (2018)
29. Lindell, Y.: An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 93–109. Springer, Heidelberg (2015)
30. Liu, Q., Zhandry, M.: Revisiting post-quantum Fiat-Shamir. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Part II, volume 11693 of LNCS, pp. 326–355. Springer, Heidelberg (2019)
31. Lombardi, A., Quach, W., Rothblum, R.D., Wichs, D., David, J.W.: New constructions of reusable designated-verifier NIZKs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Part III, volume 11694 of LNCS, pp. 670–700. Springer, Heidelberg (2019)
32. 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)
33. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC, pp. 427–437. ACM Press, May (1990)
34. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Part I, volume 11692 of LNCS, pp. 89–114. Springer, Heidelberg (2019)
35. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. Part I, volume 10401 of LNCS, pp. 570–596. Springer, Heidelberg (2017)
36. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
37. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. Part II, volume 9057 of LNCS, pp. 755–784. Springer, Heidelberg (2015)
38. Wang, X., Hongbo, Yu.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)



# Fine-Grained Verifier NIZK and Its Applications

Xiangyu Liu<sup>1,2</sup>, Shengli Liu<sup>1,2,3(✉)</sup>, Shuai Han<sup>1,2(✉)</sup>, and Dawu Gu<sup>1</sup>

<sup>1</sup> School of Electronic Information and Electrical Engineering,  
Shanghai Jiao Tong University, Shanghai 200240, China  
{xiangyu.liu, slliu, dalen17, dwgu}@sjtu.edu.cn

<sup>2</sup> State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

<sup>3</sup> Westone Cryptologic Research Center, Beijing 100070, China

**Abstract.** In this paper, we propose a new type of non-interactive zero-knowledge (NIZK), called *Fine-grained Verifier NIZK (FV-NIZK)*, which provides more flexible and more fine-grained verifiability of proofs than standard NIZK that supports public verifiability and designated-verifier NIZK (DV-NIZK) that supports private verifiability. FV-NIZK has two statistically equivalent verification approaches:

- a master verification using the master secret key  $msk$ ;
- a fine-grained verification using a derived secret key  $sk_d$ , which is derived from  $msk$  w.r.t.  $d$  (which may stand for user identity, email address, vector, etc.).

We require *unbounded simulation soundness (USS)* of FV-NIZK to hold, even if an adversary obtains derived secret keys  $sk_d$  with  $d$  of its choices, and define *proof pseudorandomness* which stipulates the pseudorandomness of proofs for adversaries that are not given any secret key.

We present two instantiations of FV-NIZK for linear subspace languages, based on the matrix decisional Diffie-Hellman (MDDH) assumption. One of the FV-NIZK instantiations is *pairing-free* and achieves almost tight USS and proof pseudorandomness.

We illustrate the usefulness of FV-NIZK by showing two applications and obtain the following pairing-free schemes:

- the *first* almost tightly multi-challenge CCA (mCCA)-secure inner-product functional encryption (IPFE) scheme *without pairings*;
- the *first* public-key encryption (PKE) scheme that reconciles the inherent contradictions between public verifiability and anonymity. We formalize such PKE as *Fine-grained Verifiable PKE (FV-PKE)*, which derives a special key from the decryption secret key, such that for those who obtain the derived key, they can check the validity of ciphertexts but the anonymity is lost from their views (CCA-security still holds for them), while for others who do not get the derived key, they cannot do the validity check but the anonymity holds for them. Our FV-PKE scheme achieves almost tight mCCA-security for adversaries who obtain the derived keys, and achieves almost tight ciphertext pseudorandomness (thus anonymity) for others who do not get any derived key.

## 1 Introduction

**NIZK with Unbounded Simulation Soundness (USS).** Over decades, non-interactive zero-knowledge (NIZK) proofs have shown great power in constructing a variety of cryptographic primitives, e.g., public-key encryption (PKE) [14,27], digital signatures [7], etc. Towards better efficiency and shorter proofs, Jutla and Roy [23] defined a weaker notion called *quasi-adaptive* NIZK (QA-NIZK), where the common reference string (CRS) might depend on the specific language. In this paper, we will focus on quasi-adaptive NIZK and omit the term “quasi-adaptive” for simplicity.

One important security property for NIZK is *unbounded simulation soundness* (USS) [25,30], which plays an important role in many applications of NIZK, e.g., CCA-secure PKE [16,21], publicly verifiable CCA identity-based encryption (IBE) [22], structure preserving signatures [4,5], etc. Loosely speaking, USS requires the computational hardness for an adversary to generate a valid proof for an instance outside the language, even if the adversary has access to an oracle that outputs simulated proofs for instances (not necessarily in the language) of its choices.

**Tight Security and NIZK with Tight USS.** The security of a cryptographic primitive is usually proved via a reduction, which turns an adversary  $\mathcal{A}$  that breaks the security of the primitive with running time  $t$  and advantage  $\epsilon$  into an algorithm  $\mathcal{B}$  that solves some hard problem with running time  $t' \approx t$  and advantage  $\epsilon'$ . Intuitively, we would desire  $\epsilon'$  to be as large as  $\epsilon$ . To reflect this, we define  $L := \epsilon/\epsilon'$  as the security loss factor, which is the smaller the better. We call the reduction *tight* if  $L$  is a small constant or *almost tight* if  $L$  is linear (or even better, logarithmic) in the security parameter  $\lambda$ . For a loose reduction,  $L$  usually depends on  $\mathcal{A}$ 's behaviours, e.g., the number of  $\mathcal{A}$ 's queries, which can be as large as  $2^{50}$  in practical settings.

Pursuing (almost) tight security has both theoretical and practical significance. For a scheme with a loose security reduction, the deployer has to choose larger security parameters to compensate the security loss, resulting in larger elements and lower efficiency. In contrast, schemes with (almost) tight security enjoy many advantages like universal key recommendations and more flexible choices of parameters. Recently, (almost) tight security has been explored in many areas, including PKE [16,17,20,21], signatures [8,19,21,24], IBE [9,11], etc.

In the scenario of NIZK, Libert et al. [25] proposed the first scheme with (almost) tight USS, and Gay et al. [16] gave a more efficient construction later. In both schemes, the size of the CRS (in terms of the number of group elements) is linear in  $\lambda$ . The first (almost) tightly secure NIZK with constant-size CRS was designed by Abe et al. [5]. Recently in [4], Abe et al. proposed a shorter NIZK with both constant-size CRS and proofs.

**Designated-Verifier NIZK (DV-NIZK).** Standard NIZK allows *public verification*, so that anyone who gets the CRS can verify the validity of proofs. Such a property is useful in certain applications, e.g., when constructing signature



schemes [4, 7], the public verifiability of signatures requires the public verifiability of NIZK proofs. However, in some other applications such as constructing CCA-secure PKE [12, 16], public verification is not necessary, and in fact, a *designated-verifier* NIZK (DV-NIZK) [16] that supports only private verification of proofs is sufficient. Roughly speaking, DV-NIZK is the same as NIZK except that, the verification algorithm additionally takes a secret key  $sk$  as input, so that only the designated verifier can check the validity of proofs. Moreover, the secret key should be kept private, since otherwise the (simulation) soundness might not hold any more.

Compared to NIZK, DV-NIZK usually has more succinct and more efficient constructions, since it is only required to support private verification. For example, the efficient hash proof systems (HPS) in [12] can be viewed as DV-NIZKs. As another example, to the best of our knowledge, all NIZK schemes with tight USS (constructed in discrete-logarithm setting) relies on bilinear pairings to support public verification [4, 5, 16, 25], while DV-NIZK with tight USS can be constructed without pairings [16].

However, both NIZK (that supports public verification) and DV-NIZK (that supports private verification) have their limitations on the flexibility of verification in certain applications. We demonstrate with two examples below.

**Fine-grained Verification Setting in IPFE.** Inner-product functional encryption (IPFE) [1] is a special subclass of functional encryption [10, 28] for inner-product functions. In an IPFE scheme, a ciphertext is an encryption of a vector  $\mathbf{x} \in \mathbb{Z}^m$ , a secret key  $\widetilde{sk}_{\mathbf{y}}$  (delegated from the master secret key  $\widetilde{msk}$ ) is related with a vector  $\mathbf{y} \in \mathbb{Z}^m$ , and the decryption just returns their inner product  $\langle \mathbf{x}, \mathbf{y} \rangle$ . The inner-product function supports a large set of computation formulas, ranging from conjunctions and disjunctions to descriptive statistics and polynomial evaluations.

There are many explorations of CPA-secure IPFE schemes over the past years, e.g., [2, 6, 31]. All ciphertexts in these constructions fall into the HPS paradigm [12] with a pattern  $(c, v)$ , where  $c$  is an instance in a language specified by the public key and  $v$  masks the message  $m$ .

To lift these CPA-secure IPFE schemes to CCA-secure IPFE schemes, one may want to resort to NIZK or DV-NIZK to reject ill-formed ciphertexts (i.e., ciphertexts with  $c$  outside the language) in decryption, thus making the decryption oracle useless to the adversary. This can be done by adding a NIZK/DV-NIZK proof in the ciphertext to prove that  $c$  belongs to the language. However, here comes the dilemma when choosing a suitable NIZK argument:

- DV-NIZK does not work in this setting with the following reason. To verify the well-formedness of ciphertexts, the decryption algorithm of IPFE has to know the secret key  $sk$  of DV-NIZK to verify the DV-NIZK proofs in ciphertexts. Thus all secret keys  $\widetilde{sk}_{\mathbf{y}}$  of IPFE should contain the secret key  $sk$ . However, note that an adversary in the CPA/CCA-security experiment of IPFE is free to ask  $\widetilde{sk}_{\mathbf{y}}$  for vectors  $\mathbf{y}$  of its choices. Consequently, the adversary only needs to ask a single  $\widetilde{sk}_{\mathbf{y}}$  to know the secret key  $sk$  of DV-

- NIZK, in which case the (simulation) soundness of DV-NIZK might not hold any more, and consequently, the CCA-security of IPFE might not hold.
- In contrast, NIZK with public verification is sufficient, but seems to be overqualified in this setting. In fact, it is not necessary for everyone, but only those who hold secret keys  $\widehat{sk}_y$ , to be able to check the well-formedness of ciphertexts in decryption.

In summary, DV-NIZK does not work in converting CPA-secure IPFE schemes into CCA-secure ones but it has more efficient constructions (e.g., pairing-free constructions), while NIZK is sufficient but at the price of heavy constructions (especially, the pairing operations) and it seems to be overqualified.

Actually, what we need is a NIZK with *fine-grained verifiability*, lying between public verifiability and private verifiability. More precisely, there is a master secret key  $msk$  for verification, and the ability of verification can be delegated via deriving different secret keys  $sk_d$  from  $msk$  w.r.t. different  $d$  (which stands for, e.g., user identity, email address, vector, etc.), so that one can use  $sk_d$  to do the verification of NIZK proofs (hence execute decryptions of IPFE). On the one hand, all these verification approaches, no matter using  $msk$  or using  $sk_d$  w.r.t. any  $d$ , are statistically equivalent. On the other hand, (simulation) soundness is guaranteed even if the adversary obtains several  $sk_d$  with  $d$  chosen by itself, as long as  $msk$  is not leaked to the adversary.

In this work, we will formalize such NIZK as *Fine-grained Verifier NIZK (FV-NIZK)*, and show that it is sufficient for lifting CPA-secure IPFE schemes to CCA-secure ones. FV-NIZK has pairing-free constructions, and hence solves the aforementioned dilemma.

**Fine-grained Verification Setting in PKE.** In traditional PKE setting, only the owner of the secret key  $sk$  can check the validity of a ciphertext (i.e., whether a ciphertext decrypts to some plaintext or the decryption fails). In some applications, it is desirable to outsource this validity check to others. For example, a manager may ask an assistant to filter out invalid ciphertexts for her/him so that the manager can decrypt only the valid ciphertexts herself/himself, but the manager does not want to reveal the secret key to the assistant. To solve such problems, the concept of *publicly verifiable* PKE (PV-PKE) [3, 21] is developed, in which anyone can check the validity of a ciphertext with only the public key of the owner.

Though public verifiability is desirable in some scenarios, it also brings the disadvantage of *losing anonymity*. Namely, anyone can identify the intended receiver of a ciphertext, by just doing a verification under someone's public key.

In order to reconcile the inherent contradictions between public verifiability and anonymity, we put forward a new primitive called *Fine-grained Verifiable PKE (FV-PKE)*, which can derive a special key (for validity check of ciphertexts) from the secret key (for decryption). Roughly speaking, with the derived key, one can check the validity of ciphertexts but cannot decrypt the ciphertexts, while without the key, the anonymity of ciphertexts holds. Let us move back to the above example. Now the manager can safely give this derived key to the assistant to filter out invalid ciphertexts. For the assistant, the anonymity is lost

but the CCA-security of the PKE still holds. For others who only obtain the public key of the manager, the anonymity of ciphertexts holds. Furthermore, we allow that different keys (for validity check) can be derived from the secret key (for decryption), to achieve fine-grained verifiability.

Now we consider how to construct FV-PKE. Let us start from any CPA-secure PKE scheme. To lift it to CCA-secure FV-PKE, one may want to resort to NIZK (as in [14, 27]) or DV-NIZK (as in [12, 16]) to reject ill-formed ciphertexts. However, neither NIZK nor DV-NIZK leads to FV-PKE:

- DV-NIZK does not support the delegation of verifiability. Thus to check the validity of ciphertexts, the derived key of PKE should contain the secret key of DV-NIZK. Then for anyone with the derived key (e.g., the assistant in the above example), the (simulation) soundness of DV-NIZK might not hold, and consequently, the CCA-security of PKE might not hold.
- NIZK allows public verification of proofs. Thus anyone (who obtains the CRS of NIZK from the public key of PKE<sup>1</sup>) can check the validity of ciphertexts, and consequently the anonymity of PKE is sacrificed. Even in the setting that all users of a group (e.g., a company or a college) share the same CRS, the identity of the group is still leaked.

In fact, our new *Fine-grained Verifier NIZK (FV-NIZK)* is suitable in this setting and can successfully convert a CPA-secure PKE into a CCA-secure FV-PKE. More precisely, the owner can derive an  $sk_d$  from the master secret key  $msk$  of FV-NIZK, so that  $sk_d$  can be used to do validity check of ciphertexts. Meanwhile, obtaining  $sk_d$  does not compromise the (simulation) soundness of FV-NIZK, and hence CCA-security of PKE holds, even for those who have the derived key. Furthermore, for others who do not obtain the derived key, the anonymity of PKE holds, as long as the underlying CPA-secure PKE is anonymous and FV-NIZK has pseudorandom proofs.

**Our Contributions.** Now we summarize our contributions in this paper. We introduce a new primitive called *Fine-grained Verifier NIZK (FV-NIZK)*, which provides more flexible and more fine-grained verifiability than standard NIZK (with public verifiability) and DV-NIZK (with private verifiability). Intuitively, FV-NIZK has two main verification approaches:

- a master verification (MVer) using the master secret key  $msk$ ;
- a fine-grained verification (FVer) using a derived secret key  $sk_d$ , which is derived from  $msk$  w.r.t.  $d \in \mathcal{D}$ . Here  $d$  belongs to a delegation space  $\mathcal{D}$ , and may stand for user identity, email address, vector, etc.

We equip FV-NIZK with a set of useful security properties. The statistical *verification equivalence* property requires that the two verification approaches, no matter using  $msk$  or using  $sk_d$  w.r.t. any  $d \in \mathcal{D}$ , are statistically equivalent. Besides, we adapt *unbounded simulation soundness (USS)* to FV-NIZK, by additionally allowing the adversary to obtain derived secret keys  $sk_d$  with  $d$  of its

<sup>1</sup> Note that the CRS of NIZK is contained in the public key of PKE, since the encryption algorithm of PKE involves NIZK proof generation which requires the CRS.

choices. We also define *proof pseudorandomness* which stipulates the pseudorandomness of proofs for adversaries that are not given any secret key.

Then we propose two instantiations of FV-NIZK with almost tight USS for linear subspace languages, based on the matrix decisional Diffie-Hellman (MDDH) assumption [15] (which covers the standard DDH and  $k$ -Linear assumptions).

- Our first instantiation is inspired by the DV-NIZK scheme constructed in [16]. The resulting FV-NIZK is *pairing-free*, and achieves almost tight USS and proof pseudorandomness, with a linear loss factor  $L = O(\lambda)$ .
- Our second instantiation is inspired by the DV-NIZK and NIZK schemes in [4]. The resulting FV-NIZK is pairing-based, but involves *less pairing operations* than the NIZK scheme in [4]. It achieves almost tight USS with a loss factor  $L = O(\log \lambda)$ , logarithmic in the security parameter  $\lambda$ .

Finally, we illustrate the usefulness of FV-NIZK by showing two applications.

- The first application is in constructing CCA-secure IPFE. Using our FV-NIZK with almost tight USS as the core technique tool, we construct a tightly multi-challenge CCA (mCCA)-secure IPFE scheme from the almost tightly multi-challenge CPA (mCPA)-secure IPFE proposed in [31].

By instantiating FV-NIZK, we obtain the first almost tightly mCCA-secure IPFE scheme *without pairings*, where the loss factor is  $L = O(\lambda)$ . We also obtain another almost tightly mCCA-secure IPFE scheme that uses less pairing operations than the only known scheme [26] (*12 vs.  $2m + 16$  pairings*, with  $m$  the vector dimension of IPFE), where the loss factor is  $L = O(\log \lambda)$ , the same as [26].

- The second application is in constructing *Fine-grained Verifiable PKE (FV-PKE)*. This is a new primitive formalized in this paper to reconcile the inherent contradictions between public verifiability and anonymity of PKE. Loosely speaking, FV-PKE derives a special key from the decryption secret key, such that for those who obtain the derived key, they can check the validity of ciphertexts but the anonymity is lost from their views (CCA-security still holds for them), while for others who do not get the derived key, they cannot do the validity check but the anonymity holds for them.

By using our first FV-NIZK instantiation with almost tight USS and proof pseudorandomness as the core building block, we construct the first FV-PKE scheme that achieves both almost tight mCCA-security and almost tight ciphertext pseudorandomness (thus anonymity). Moreover, the FV-PKE scheme is pairing-free.

**Technical Overview of Our FV-NIZK Instantiations.** Below we give a high-level overview of our FV-NIZK instantiations from the MDDH assumption. Let  $\mathbb{G}$  be a cyclic group of order  $q$  with generator  $g$ . For a matrix  $\mathbf{A} := (a_{ij}) \in \mathbb{Z}_q^{n_1 \times n_2}$ , we define  $[\mathbf{A}] := (g^{a_{ij}}) \in \mathbb{G}^{n_1 \times n_2}$  as the implicit representation of  $\mathbf{A}$  in  $\mathbb{G}$  [15]. Our FV-NIZK instantiations are for linear subspace language  $\mathcal{L}_{[\mathbf{A}]} := \text{Span}([\mathbf{A}]) := \{\mathbf{c} \in \mathbb{G}^{n_1} \mid \exists \mathbf{s} \text{ s.t. } \mathbf{c} = \mathbf{A}\mathbf{s}\}$  and the delegation space is  $\mathcal{D} := \mathbb{Z}_q^m$ .

Our starting point is the tag-based DV-NIZK scheme proposed by Gay et al. [16], which is pairing-free and has almost tight USS, as recalled below. The CRS is  $\text{crs} := ([\mathbf{k}^\top \mathbf{A}], [\mathbf{B}], \{\widehat{\mathbf{k}}_{\ell,b}^\top \mathbf{B}\}_{\ell,b})$ , and the secret key  $msk$  for verification is  $msk := (\mathbf{k}, \{\widehat{\mathbf{k}}_{\ell,b}\}_{\ell,b})$ , where  $\mathbf{k} \xleftarrow{\$} \mathbb{Z}_q^{n_1}$ ,  $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{3k \times k}$  and  $\widehat{\mathbf{k}}_{\ell,b} \xleftarrow{\$} \mathbb{Z}_q^{3k}$  for  $1 \leq \ell \leq \lambda, b \in \{0, 1\}$ . With respect to a tag  $\tau \in \{0, 1\}^\lambda$ , the proof of  $[\mathbf{c}] = [\mathbf{A}]\mathbf{s} \in \mathcal{L}_{[\mathbf{A}]}$  is  $\pi := ([\mathbf{t}], [u])$ , where  $[\mathbf{t}] := [\mathbf{B}]\mathbf{r}$  for  $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k$  and

$$[u] := [\mathbf{k}^\top \mathbf{A}]\mathbf{s} + [\widehat{\mathbf{k}}_\tau^\top \mathbf{B}]\mathbf{r}, \quad \text{with } \widehat{\mathbf{k}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{k}}_{\ell,\tau_\ell},$$

which can be verified via  $[u] \stackrel{?}{=} \mathbf{k}^\top [\mathbf{c}] + \widehat{\mathbf{k}}_\tau^\top [\mathbf{t}]$  using  $msk$ .

*How to derive keys for fine-grained verification?* To support deriving keys for different delegations  $\mathbf{d} \in \mathcal{D} = \mathbb{Z}_q^m$ , a natural idea is to extend the master secret key in the DV-NIZK above from a set of vectors to a sets of matrices, i.e.,  $\text{crs} := ([\mathbf{KA}], [\mathbf{B}], \{\widehat{\mathbf{K}}_{\ell,b} \mathbf{B}\}_{\ell,b})$  and  $msk := (\mathbf{K}, \{\widehat{\mathbf{K}}_{\ell,b}\}_{\ell,b})$  with  $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_q^{m \times n_1}$  and  $\widehat{\mathbf{K}}_{\ell,b} \xleftarrow{\$} \mathbb{Z}_q^{m \times 3k}$ . Accordingly, the proof is  $\pi := ([\mathbf{t}], [\mathbf{u}])$  with

$$[\mathbf{u}] := [\mathbf{KA}]\mathbf{s} + [\widehat{\mathbf{K}}_\tau \mathbf{B}]\mathbf{r}, \quad \text{with } \widehat{\mathbf{K}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{K}}_{\ell,\tau_\ell},$$

and the *master* verification checks  $[\mathbf{u}] \stackrel{?}{=} \mathbf{K}[\mathbf{c}] + \widehat{\mathbf{K}}_\tau [\mathbf{t}]$  using  $msk$ . One can view it as  $m$ -parallel DV-NIZKs in [16].

Now we can derive a key  $sk_{\mathbf{d}}$  w.r.t. a delegation  $\mathbf{d} \in \mathcal{D} = \mathbb{Z}_q^m$  as follows

$$sk_{\mathbf{d}} := (\mathbf{d}, \mathbf{d}^\top \mathbf{K}, \{\mathbf{d}^\top \widehat{\mathbf{K}}_{\ell,b}\}_{\ell,b}),$$

and the *fine-grained* verification using  $sk_{\mathbf{d}}$  checks

$$\mathbf{d}^\top [\mathbf{u}] \stackrel{?}{=} \mathbf{d}^\top \mathbf{K}[\mathbf{c}] + \mathbf{d}^\top \widehat{\mathbf{K}}_\tau [\mathbf{t}].$$

Intuitively, delegation algorithm for  $\mathbf{d}$  derives a “projection” of  $msk$  on  $\mathbf{d}$ , so that this derived secret key can be used to check the proof on  $\mathbf{d}$ ’s projection.

However, here come two problems. Firstly, the two verification approaches are not statistically equivalent. In fact, given only  $\text{crs}$ , an adversary  $\mathcal{A}$  can easily produce a proof  $\pi^* = ([\mathbf{t}^*], [\mathbf{u}^*])$  for  $[\mathbf{c}]$  such that it passes the fine-grained verification w.r.t.  $sk_{\mathbf{d}}$ , but does not pass the master verification, i.e.,

$$\mathbf{d}^\top [\mathbf{u}^*] = \mathbf{d}^\top \mathbf{K}[\mathbf{c}] + \mathbf{d}^\top \widehat{\mathbf{K}}_\tau [\mathbf{t}^*], \quad \text{but } [\mathbf{u}^*] \neq \mathbf{K}[\mathbf{c}] + \widehat{\mathbf{K}}_\tau [\mathbf{t}^*].$$

This can be done as follows.  $\mathcal{A}$  first generates a proof  $\pi = ([\mathbf{t}], [\mathbf{u}])$  for an instance  $[\mathbf{c}] \in \mathcal{L}_{[\mathbf{A}]}$  honestly using  $\text{crs}$ , and then chooses a pair of non-zero orthogonal vectors  $\mathbf{d}, \mathbf{e} \in \mathbb{Z}_q^m$  s.t.  $\mathbf{d}^\top \mathbf{e} = 0$ , and sets  $\pi^* = ([\mathbf{t}^*], [\mathbf{u}^*]) := ([\mathbf{t}], [\mathbf{u} + \mathbf{e}])$ . Clearly  $[\mathbf{u}^*] - \mathbf{K}[\mathbf{c}] - \widehat{\mathbf{K}}_\tau [\mathbf{t}^*] = [\mathbf{u}^*] - [\mathbf{u}] = [\mathbf{e}] \neq [\mathbf{0}]$ , but  $\mathbf{d}^\top ([\mathbf{u}^*] - \mathbf{K}[\mathbf{c}] - \widehat{\mathbf{K}}_\tau [\mathbf{t}^*]) = \mathbf{d}^\top [\mathbf{e}] = [0]$ .

Moreover, USS cannot hold if an adversary  $\mathcal{A}$  is allowed to obtain derived keys. Due to the linearity of  $sk_{\mathbf{d}}$  in  $\mathbf{d}$ , each derived key  $sk_{\mathbf{d}}$  leaks a part of information about  $msk$ . If  $\mathcal{A}$  asks derived keys for  $m$  linearly independent vectors  $\mathbf{d}$ , then the whole  $msk$  is exposed to  $\mathcal{A}$ , and consequently,  $\mathcal{A}$  can easily generate a valid proof for an instance  $[c] \notin \mathcal{L}_{[\mathcal{A}]}$  via computing  $[\mathbf{u}] := \mathbf{K}[c] + \widehat{\mathbf{K}}_{\tau}[\mathbf{t}]$ .

*First Idea. Introducing a Random Matrix as a Secret Permutation.* In order to solve the aforementioned problems, we introduce a uniformly random matrix  $\mathbf{M} \in \mathbb{Z}_q^{m \times m}$  in  $msk$ , i.e.,  $msk := (\mathbf{K}, \{\widehat{\mathbf{K}}_{\ell,b}\}_{\ell,b}, \mathbf{M})$  with  $\mathbf{M} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{m \times m}$ . The crs, the proof generation and the master verification approach are the same as before, while the key deriving process and fine-grained verification are changed as follows. Now the derived key  $sk_{\mathbf{d}}$  w.r.t.  $\mathbf{d} \in \mathbb{Z}_q^m$  is

$$sk_{\mathbf{d}} := (\mathbf{d}^{\top} \mathbf{M}, \mathbf{d}^{\top} \mathbf{M} \mathbf{K}, \{\mathbf{d}^{\top} \mathbf{M} \widehat{\mathbf{K}}_{\ell,b}\}_{\ell,b}),$$

and the *fine-grained* verification using  $sk_{\mathbf{d}}$  checks

$$\mathbf{d}^{\top} \mathbf{M} [\mathbf{u}] \stackrel{?}{=} \mathbf{d}^{\top} \mathbf{M} \mathbf{K} [c] + \mathbf{d}^{\top} \mathbf{M} \widehat{\mathbf{K}}_{\tau} [\mathbf{t}].$$

Intuitively, now the  $sk_{\mathbf{d}}$  no longer projects  $msk$  on vector  $\mathbf{d}$ , but on a random vector  $\mathbf{d}^{\top} \mathbf{M}$  which secretly rotates  $\mathbf{d}$  by the matrix  $\mathbf{M}$  in  $msk$ . As long as  $\mathbf{d}^{\top} \mathbf{M}$  contains enough entropy from an adversary  $\mathcal{A}$ 's view<sup>2</sup>, it is impossible for  $\mathcal{A}$  to output a proof  $\pi^* = ([\mathbf{t}^*], [\mathbf{u}^*])$  for  $[c]$  such that

$$\mathbf{d}^{\top} \mathbf{M} [\mathbf{u}^*] = \mathbf{d}^{\top} \mathbf{M} \mathbf{K} [c] + \mathbf{d}^{\top} \mathbf{M} \widehat{\mathbf{K}}_{\tau} [\mathbf{t}^*], \text{ but } [\mathbf{u}^*] \neq \mathbf{K} [c] + \widehat{\mathbf{K}}_{\tau} [\mathbf{t}^*],$$

except with negligible probability, since otherwise  $[\mathbf{u}^*] - \mathbf{K} [c] - \widehat{\mathbf{K}}_{\tau} [\mathbf{t}^*]$  constitutes a non-zero vector in the right kernel space of  $\mathbf{d}^{\top} \mathbf{M}$ . As a result, verification equivalence is guaranteed.

However, USS still cannot hold, since the whole  $msk$  is still exposed to  $\mathcal{A}$  if  $\mathcal{A}$  asks derived keys for  $m$  linearly independent vectors  $\mathbf{d}$ .

*Second Idea. Enlarging the Random Matrix as an Entropy Filter.* To rescue USS, we enlarge  $\mathbf{M}$  to be a matrix in  $\mathbb{Z}_q^{m \times (m+1)}$ . Now even if  $\mathcal{A}$  queries derived keys  $sk_{\mathbf{d}}$  for  $m$  linearly independent vectors  $\mathbf{d}$ , the information about  $msk$  leaked to  $\mathcal{A}$  is limited in

$$(\mathbf{M}, \mathbf{M} \mathbf{K}, \{\mathbf{M} \widehat{\mathbf{K}}_{\ell,b}\}_{\ell,b}),$$

and there is still entropy left. More precisely, let  $\mathbf{m}^{\perp} \in \mathbb{Z}_q^{m+1}$  be a vector s.t.  $\mathbf{M} \mathbf{m}^{\perp} = \mathbf{0}$ , and let  $(\mathbf{K}, \{\widehat{\mathbf{K}}_{\ell,b}\}_{\ell,b}) := (\mathbf{K}' + \mathbf{m}^{\perp} \boxed{\widetilde{\mathbf{k}}}, \{\widehat{\mathbf{K}}'_{\ell,b} + \mathbf{m}^{\perp} \boxed{\widetilde{\mathbf{k}}_{\ell,b}}\}_{\ell,b})$ , where

<sup>2</sup> This entropy requirement is necessary to achieve verification equivalence, see Remark 1 in Sect. 3 for more discussions.

$\mathbf{K}' \xleftarrow{\$} \mathbb{Z}_q^{m \times n_1}$ ,  $\widehat{\mathbf{K}}'_{\ell,b} \xleftarrow{\$} \mathbb{Z}_q^{m \times 3k}$  and  $\boxed{\widetilde{\mathbf{k}} \xleftarrow{\$} \mathbb{Z}_q^{1 \times n_1}, \widetilde{\mathbf{k}}_{\ell,b} \xleftarrow{\$} \mathbb{Z}_q^{1 \times 3k}}$ . Then the entropy of  $\boxed{(\widetilde{\mathbf{k}}, \{\widetilde{\mathbf{k}}_{\ell,b}\}_{\ell,b})}$  is reserved from the derived key queries, by observing that

$$(\mathbf{M}, \mathbf{MK}, \{\widehat{\mathbf{MK}}_{\ell,b}\}_{\ell,b}) = (\mathbf{M}, \mathbf{MK}', \{\widehat{\mathbf{MK}}'_{\ell,b}\}_{\ell,b}).$$

Consequently, the enlarged matrix  $\mathbf{M}$  also works as an entropy filter in our FV-NIZK instantiation.

Finally, by using the reserved  $\boxed{(\widetilde{\mathbf{k}}, \{\widetilde{\mathbf{k}}_{\ell,b}\}_{\ell,b})}$  (which in turn corresponds to the *msk* of the DV-NIZK in [16]), we can prove the almost tight USS of our FV-NIZK following the proof strategy in [16].

*Others.* By using the MDDH assumption, we further prove the almost tight pseudorandomness of the proofs  $\pi = ([\mathbf{t}], [\mathbf{u}])$  for adversaries that are not given any derived secret key. This property serves as the core technical tool to achieve anonymity in the fine-grained verifiable PKE application.

Moreover, we note that our aforementioned ideas seem to be general ideas to lift a DV-NIZK scheme with good linearity to an FV-NIZK. Following the similar ideas, we also extend the DV-NIZK scheme proposed by Abe et al. [4] to an FV-NIZK, as our second instantiation.

**Roadmap.** In Sect. 2 we present notations and recall the MDDH assumptions. The definition and security properties of FV-NIZK are formally described in Sect. 3. Then in Sect. 4, we propose two instantiations of FV-NIZK with almost tight USS for linear subspace languages. In Sect. 5, we illustrate two applications of FV-NIZK in IPFE and FV-PKE, respectively.

## 2 Preliminaries

Let  $\lambda \in \mathbb{N}$  denote the security parameter and  $\emptyset$  the empty set. For  $\mu \in \mathbb{N}$ , define  $[\mu] := \{1, 2, \dots, \mu\}$ . For  $a, b \in \mathbb{Z}$  with  $a < b$ , define  $[a, b] := \{a, a + 1, \dots, b\}$ . Denote by  $x := y$  the operation of assigning  $y$  to  $x$ . Denote by  $x \xleftarrow{\$} \mathcal{Q}$  the operation of sampling  $x$  uniformly at random from a set  $\mathcal{Q}$ . For a distribution  $\mathcal{D}$ , denote by  $x \leftarrow \mathcal{D}$  the operation of sampling  $x$  according to  $\mathcal{D}$ . For an algorithm  $\mathcal{A}$ , denote by  $y \leftarrow \mathcal{A}(x; r)$ , or simply  $y \leftarrow \mathcal{A}(x)$ , the operation of running  $\mathcal{A}$  with input  $x$  and randomness  $r$  and assigning the output to  $y$ . ‘‘PPT’’ is short for probabilistic polynomial-time.  $\text{poly}(\lambda)$  and  $\text{negl}(\lambda)$  denote polynomial and negligible functions in  $\lambda$ , respectively.

We use bold lower-case letters to denote vectors (e.g.,  $\mathbf{x}$ ), and bold upper-case letters to denote matrices (e.g.,  $\mathbf{A}$ ). Unless specific description, all vectors are column vectors in this paper. For matrices  $\mathbf{A}$  and  $\mathbf{B}$ , we use  $\mathbf{A} \otimes \mathbf{B}$  for their tensor (or Kronecker) product  $(a_{i,j} \mathbf{B})_{i,j}$ . For vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^m$ , let  $\langle \mathbf{x}, \mathbf{y} \rangle$  denote their inner product  $\mathbf{x}^\top \mathbf{y} \in \mathbb{Z}$ . Let  $\mathbf{I}_n$  and  $\mathbf{0}_{n_1 \times n_2}$  denote the identity and zero matrices respectively.

For random variables  $X$  and  $Y$ , the min-entropy of  $X$  is defined as  $\mathbf{H}_\infty(X) := -\log(\max_x \Pr[X = x])$ , and the average min-entropy of  $X$  conditioned on  $Y$  is defined as  $\widetilde{\mathbf{H}}_\infty(X|Y) := -\log(\mathbb{E}_{y \leftarrow Y}[\max_x \Pr[X = x|Y = y]])$ , following [13].

**Definition 1 (Collision Resistant Hash Families).** *Let  $\mathcal{X}, \mathcal{Y}$  be two finite sets. A family of hash functions  $\mathcal{H} = \{H : \mathcal{X} \rightarrow \mathcal{Y}\}$  is collision resistant, if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{\mathcal{H}, \mathcal{A}}^{\text{cr}}(\lambda) := \Pr[H \xleftarrow{\$} \mathcal{H}, (x, x') \leftarrow \mathcal{A}(H) : x \neq x' \wedge H(x) = H(x')] \leq \text{negl}(\lambda).$$

## 2.1 Group Assumptions

Let  $\mathcal{G} = (\mathbb{G}, g, q) \leftarrow \text{GGen}$  be a group generation algorithm that inputs  $1^\lambda$  and returns a cyclic group  $\mathbb{G}$  of order  $q$  with generator  $g$ . For matrix  $\mathbf{A} := (a_{ij})_{n_1 \times n_2}$  with  $a_{ij} \in \mathbb{Z}_q$ , we define  $[\mathbf{A}] := (g^{a_{ij}})_{n_1 \times n_2}$  as the implicit representation of  $\mathbf{A}$  in  $\mathbb{G}$  [15]. For  $\mathbf{A} \in \mathbb{Z}_q^{n_1 \times n_2}$ , the linear subspace spanned by  $\mathbf{A}$  is  $\text{Span}(\mathbf{A}) := \{\mathbf{c} \mid \exists \mathbf{s} \text{ s.t. } \mathbf{c} = \mathbf{A}\mathbf{s}\}$ , and similarly,  $\text{Span}([\mathbf{A}]) := \{\mathbf{c} \mid \exists \mathbf{s} \text{ s.t. } \mathbf{c} = \mathbf{A}\mathbf{s}\}$ . Given  $\mathbf{A} \in \mathbb{Z}_q^{n_1 \times n_2}$ , it is efficient to sample an  $\mathbf{A}^\perp \in \mathbb{Z}_q^{(n_1 - n_2) \times n_1}$  s.t.  $\mathbf{A}^\perp \mathbf{A} = \mathbf{0}$ .

Let  $\ell, k \in \mathbb{N}$  and  $\ell > k$ . A matrix distribution  $\mathcal{D}_{\ell, k}$  is a probabilistic distribution that outputs matrices in  $\mathbb{Z}_q^{\ell \times k}$  of full rank  $k$  in polynomial time. Especially, if  $\mathcal{D}_{\ell, k}$  is a uniform distribution, then we denote it by  $\mathcal{U}_{\ell, k}$ . In the case  $\ell = k + 1$ , we simply denote it as  $\mathcal{D}_k$  or  $\mathcal{U}_k$ .

**Definition 2 ( $\mathcal{D}_{\ell, k}$ -MDDH Assumption).** *Let  $\mathcal{D}_{\ell, k}$  be a matrix distribution. The  $\mathcal{D}_{\ell, k}$ -Matrix Decisional Diffie-Hellman ( $\mathcal{D}_{\ell, k}$ -MDDH) assumption holds in  $\mathbb{G}$ , if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{\mathcal{D}_{\ell, k}, \mathbb{G}, \mathcal{A}}^{\text{mddh}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{A}\mathbf{s}]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{u}]) = 1]| \leq \text{negl}(\lambda),$$

where  $\mathcal{G} \leftarrow \text{GGen}(1^\lambda)$ ,  $\mathbf{A} \leftarrow \mathcal{D}_{\ell, k}$ ,  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$ , and  $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^\ell$ .

**Definition 3 ( $n$ -fold  $\mathcal{D}_{\ell, k}$ -MDDH Assumption).** *Let  $n \geq 1$  and let  $\mathcal{D}_{\ell, k}$  be a matrix distribution. The  $n$ -fold  $\mathcal{D}_{\ell, k}$ -MDDH assumption holds in  $\mathbb{G}$ , if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{\mathcal{D}_{\ell, k}, \mathbb{G}, \mathcal{A}}^{n\text{-mddh}} := |\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{A}\mathbf{S}]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{U}]) = 1]| \leq \text{negl}(\lambda),$$

where  $\mathcal{G} \leftarrow \text{GGen}(1^\lambda)$ ,  $\mathbf{A} \leftarrow \mathcal{D}_{\ell, k}$ ,  $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{k \times n}$ , and  $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{\ell \times n}$ .

**Lemma 1 (Random Self-Reducibility [15, 16]).** *Let  $n \geq 1$ . For any adversary  $\mathcal{A}$ , there exists an algorithm  $\mathcal{B}$  s.t.  $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + n \cdot \text{poly}(\lambda)$ , and  $\text{Adv}_{\mathcal{D}_{\ell, k}, \mathbb{G}, \mathcal{A}}^{n\text{-mddh}}(\lambda) \leq (\ell - k) \text{Adv}_{\mathcal{D}_{\ell, k}, \mathbb{G}, \mathcal{B}}^{\text{mddh}}(\lambda) + \frac{1}{q-1}$ .*

*For any adversary  $\mathcal{A}$ , there exists an algorithm  $\mathcal{B}$  s.t.  $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + n \cdot \text{poly}(\lambda)$ , and  $\text{Adv}_{\mathcal{U}_{\ell, k}, \mathbb{G}, \mathcal{A}}^{n\text{-mddh}}(\lambda) \leq \text{Adv}_{\mathcal{U}_{\ell, k}, \mathbb{G}, \mathcal{B}}^{\text{mddh}}(\lambda) + \frac{1}{q-1}$ .*

**Lemma 2 ( $\mathcal{D}_{\ell, k}$ -MDDH  $\Rightarrow \mathcal{U}_k$ -MDDH  $\Leftrightarrow \mathcal{U}_{\ell, k}$ -MDDH [15, 16]).** *Let  $\ell, k \in \mathbb{N}$  and  $\ell > k$ . For any adversary  $\mathcal{A}$ , there exists an algorithm  $\mathcal{B}$  s.t.  $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A})$ , and  $\text{Adv}_{\mathcal{U}_k, \mathbb{G}, \mathcal{B}}^{\text{mddh}}(\lambda) \leq \text{Adv}_{\mathcal{D}_{\ell, k}, \mathbb{G}, \mathcal{A}}^{\text{mddh}}(\lambda)$ .*

*For any adversary  $\mathcal{A}$ , there exists an algorithm  $\mathcal{B}$  (and vice versa) s.t.  $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A})$ , and  $\text{Adv}_{\mathcal{U}_k, \mathbb{G}, \mathcal{A}}^{\text{mddh}}(\lambda) = \text{Adv}_{\mathcal{U}_{\ell, k}, \mathbb{G}, \mathcal{B}}^{\text{mddh}}(\lambda)$ .*



### 3 Fine-Grained Verifier NIZK: Definition and Security

In this section, we give the formal definition of *Fine-grained Verifier NIZK (FV-NIZK)*, and propose a set of useful security properties for it.

Let  $\mathcal{L} = \{\mathcal{L}_\rho\}$  be a collection of NP-languages indexed by parameter  $\rho$ . Each language  $\mathcal{L}_\rho$  is determined by a binary relation  $R_\rho$ , such that an instance  $c$  belongs to  $\mathcal{L}_\rho$  iff there exists a witness  $w$  s.t.  $R_\rho(c, w) = 1$ . We consider  $\mathcal{L}_\rho$  with a trapdoor  $\text{td}_\rho$ , which can be used to decide the membership of  $\mathcal{L}_\rho$  efficiently.

**Definition 4 (Tag-Based FV-NIZK).** *A tag-based Fine-grained Verifier quasi-adaptive Non-Interactive Zero-Knowledge (FV-NIZK) argument consists of seven PPT algorithms, namely  $\Pi = (\text{Par}, \text{Gen}, \text{Prove}, \text{MVer}, \text{Sim}, \text{Delegate}, \text{FVer})$ .*

- $\text{pp} \leftarrow \text{Par}(1^\lambda, \mathcal{L}_\rho)$ . Initialization algorithm takes the security parameter  $\lambda$  and a language  $\mathcal{L}_\rho$  as inputs, and outputs a public parameter  $\text{pp}$ , which defines the tag space  $\mathcal{T}$  and the delegation space  $\mathcal{D}$ .
- $(\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ . Generation algorithm takes  $\text{pp}$  as input, and outputs a common reference string  $\text{crs}$ , a trapdoor  $\text{td}$ , and a master secret key  $\text{msk}$ . Without loss of generality, we assume  $\text{crs}$  contains  $\text{pp}$ , and it serves as an implicit input of  $\text{MVer}$ ,  $\text{Sim}$ ,  $\text{Delegate}$ , and  $\text{FVer}$ .
- $\pi \leftarrow \text{Prove}(\text{crs}, c, w, \tau)$ . Proof algorithm takes  $\text{crs}$ , an instance  $c \in \mathcal{L}_\rho$  along with a witness  $w$ , and a tag  $\tau \in \mathcal{T}$  as inputs, and outputs a proof  $\pi$ .
- $0/1 \leftarrow \text{MVer}(\text{msk}, c, \tau, \pi)$ . Master verification algorithm takes  $\text{msk}$ , an instance  $c$ , a tag  $\tau \in \mathcal{T}$  and a proof  $\pi$  as inputs, and outputs a decision bit.
- $\pi \leftarrow \text{Sim}(\text{td}, c, \tau)$ . Simulation algorithm takes  $\text{td}$ , an instance  $c$  and a tag  $\tau \in \mathcal{T}$  as inputs, and outputs a simulated proof  $\pi$ .
- $\text{sk}_d \leftarrow \text{Delegate}(\text{msk}, d)$ . Delegation algorithm takes  $\text{msk}$  and a delegation  $d \in \mathcal{D}$  as inputs, and outputs a delegated secret key  $\text{sk}_d$ .
- $0/1 \leftarrow \text{FVer}(\text{sk}_d, c, \tau, \pi)$ . Fine-grained verification algorithm takes  $\text{sk}_d$ , an instance  $c$ , a tag  $\tau \in \mathcal{T}$  and a proof  $\pi$  as inputs, and outputs a decision bit.

If the tag space  $\mathcal{T}$  is the empty set  $\emptyset$  or contains only one element (e.g.,  $\{0\}$ ), we call  $\Pi$  an FV-NIZK argument.

We require  $\Pi$  to have completeness and (perfect) zero-knowledge.

**Completeness.** For all  $\text{pp} \leftarrow \text{Par}(1^\lambda, \mathcal{L}_\rho)$ ,  $(\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ ,  $(c, w)$  s.t.  $R_\rho(c, w) = 1$ ,  $\tau \in \mathcal{T}$  and  $\pi \leftarrow \text{Prove}(\text{crs}, c, w, \tau)$ , it holds that

- (1)  $\text{MVer}(\text{msk}, c, \tau, \pi) = 1$ , and
- (2)  $\text{FVer}(\text{sk}_d, c, \tau, \pi) = 1$  for all  $\text{sk}_d \leftarrow \text{Delegate}(\text{msk}, d)$  of all  $d \in \mathcal{D}$ .

**Perfect Zero-Knowledge.** For all  $\text{pp} \leftarrow \text{Par}(1^\lambda, \mathcal{L}_\rho)$ ,  $(\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ ,  $(c, w)$  s.t.  $R_\rho(c, w) = 1$  and  $\tau \in \mathcal{T}$ , the following two distributions are identical:

$$\text{Prove}(\text{crs}, c, w, \tau) \equiv \text{Sim}(\text{td}, c, \tau).$$

Note that the first five algorithms (Par, Gen, Prove, MVer, Sim) of FV-NIZK basically constitute a DV-NIZK scheme as defined in [16]. Moreover, the two additional algorithms (Delegate, FVer) provide the fine-grained verification ability, by allowing different users owning different secret keys  $sk_d$  ( $d \in \mathcal{D}$ ) to verify proofs in different ways by invoking  $\text{FVer}(sk_d, \cdot, \cdot, \cdot)$ .

Now, we define a statistical property called *verification equivalence* for FV-NIZK. Intuitively, it requires that all proofs passing the master verification algorithm MVer using  $msk$  also pass the fine-grained verification algorithm FVer using any secret key  $sk_d$  of any  $d$ , and (with high probability) vice versa.

**Definition 5 (Verification Equivalence).** *Let  $\delta, \epsilon > 0$ . A tag-based FV-NIZK  $\Pi$  has  $(\delta, \epsilon)$ -verification equivalence, if the following two properties hold.*

1.  $\text{MVer} \implies \text{FVer}$ : For all  $\text{pp} \leftarrow \text{Par}(1^\lambda, \mathcal{L}_\rho)$ ,  $(\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ , instances  $c$ , proofs  $\pi$  and tags  $\tau \in \mathcal{T}$ , if  $\text{MVer}(\text{msk}, c, \tau, \pi) = 1$  holds, then  $\text{FVer}(sk_d, c, \tau, \pi) = 1$  holds for all  $sk_d \leftarrow \text{Delegate}(\text{msk}, d)$  of all  $d \in \mathcal{D}$ .
2.  $\text{MVer} \stackrel{w.h.p.}{\longleftarrow} \text{FVer}$ : For any (even unbounded) adversary  $\mathcal{A}$ , it holds that

$$\text{Adv}_{\Pi, \mathcal{A}, \delta}^{\text{ver-equ}}(\lambda) := \Pr[\text{Exp}_{\Pi, \mathcal{A}, \delta}^{\text{ver-equ}}(\lambda) \Rightarrow 1] \leq \epsilon,$$

where the experiment  $\text{Exp}_{\Pi, \mathcal{A}, \delta}^{\text{ver-equ}}(\lambda)$  is defined in Fig. 1.

$\text{Exp}_{\Pi, \mathcal{A}, \delta}^{\text{ver-equ}}(\lambda)$ : $\text{pp} \leftarrow \text{Par}(1^\lambda, \mathcal{L}_\rho)$ , $(\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ $\mathcal{Q}_{\text{sim}} := \emptyset$ , $\mathcal{Q}_{\text{sk}} := \emptyset$ $(c^*, \tau^*, \pi^*, d^*) \leftarrow \mathcal{A}^{\text{SIM}(\cdot, \cdot), \text{DELEGATE}(\cdot)}(\text{pp}, \text{crs})$ $sk_{d^*} \leftarrow \text{Delegate}(\text{msk}, d^*)$  If $\tilde{\mathbf{H}}_\infty(sk_{d^*}   \text{crs}, \mathcal{Q}_{\text{sim}}, \mathcal{Q}_{\text{sk}}, d^*) > \delta$ $\wedge \text{FVer}(sk_{d^*}, c^*, \tau^*, \pi^*) = 1$ $\wedge \text{MVer}(\text{msk}, c^*, \tau^*, \pi^*) = 0$ : output 1 Otherwise: output 0	$\text{SIM}(c, \tau)$ : $\pi \leftarrow \text{Sim}(\text{td}, c, \tau)$ $\mathcal{Q}_{\text{sim}} := \mathcal{Q}_{\text{sim}} \cup \{(c, \tau, \pi)\}$ Return $\pi$  $\text{DELEGATE}(d)$ : $sk_d \leftarrow \text{Delegate}(\text{msk}, d)$ $\mathcal{Q}_{\text{sk}} := \mathcal{Q}_{\text{sk}} \cup \{(d, sk_d)\}$ Return $sk_d$
--	--

**Fig. 1.** The verification equivalence experiment  $\text{Exp}_{\Pi, \mathcal{A}, \delta}^{\text{ver-equ}}(\lambda)$  for tag-based FV-NIZK. In the condition “ $\tilde{\mathbf{H}}_\infty(sk_{d^*} | \text{crs}, \mathcal{Q}_{\text{sim}}, \mathcal{Q}_{\text{sk}}, d^*)$ ”,  $sk_{d^*}$  means the distribution  $\text{Delegate}(\text{msk}, d^*; r)$  with uniformly chosen randomness  $r$ , rather than a fixed value.

*Remark 1 (On the formalization of “ $\text{MVer} \stackrel{w.h.p.}{\longleftarrow} \text{FVer}$ ”).* We stress that we do not require MVer and FVer perform identically on all inputs. In other words, there might exist  $(c, \tau, \pi)$  such that  $\text{FVer}(sk_d, c, \tau, \pi) = 1$  for some  $sk_d$  but  $\text{MVer}(\text{msk}, c, \tau, \pi) = 0$ . Similarly, for different  $d_1, d_2$ , FVer using  $sk_{d_1}$  and FVer using  $sk_{d_2}$  might perform differently on some inputs, i.e., there might exist  $(c, \tau, \pi)$  such that  $\text{FVer}(sk_{d_1}, c, \tau, \pi) = 1$  but  $\text{FVer}(sk_{d_2}, c, \tau, \pi) = 0$ .

In fact, what our “MVer  $\stackrel{w,h.p.}{\Leftarrow}$  FVer” property tries to characterize is that for any (unbounded) adversary  $\mathcal{A}$  who does not get enough information about  $sk_{d^*}$  (and thus  $msk$ ), it is hard to find a  $(c^*, \tau^*, \pi^*)$  that makes MVer and FVer perform differently. This also explains the condition “ $\tilde{H}_\infty(sk_{d^*} | \text{crs}, \mathcal{Q}_{sim}, \mathcal{Q}_{sk}, d^*) > \delta$ ” in Fig. 1 for  $\mathcal{A}$  to win. Otherwise, if the min-entropy of  $sk_{d^*}$  is lower than some threshold (say  $\delta$ ),  $\mathcal{A}$  can guess  $sk_{d^*}$  correctly with a noticeable probability. Meanwhile, it can obtain  $sk_d$  for some  $d \neq d^*$  by querying DELEGATE( $d$ ). With the knowledge of  $sk_{d^*}$  and  $sk_d$ , it is feasible for  $\mathcal{A}$  to find  $(c^*, \tau^*, \pi^*)$  such that  $\text{FVer}(sk_{d^*}, c^*, \tau^*, \pi^*) = 1$  but  $\text{FVer}(sk_d, c^*, \tau^*, \pi^*) = 0$  (e.g., via brute-force search). According to the first property “MVer  $\implies$  FVer”,  $\text{FVer}(sk_d, c^*, \tau^*, \pi^*) = 0$  implies  $\text{MVer}(msk, c^*, \tau^*, \pi^*) = 0$ , and consequently  $\mathcal{A}$  wins in  $\text{Exp}_{\Pi, \mathcal{A}, \delta}^{\text{ver-equ}}(\lambda)$ . To prevent such trivial attacks, we require  $\tilde{H}_\infty(sk_{d^*} | \text{crs}, \mathcal{Q}_{sim}, \mathcal{Q}_{sk}, d^*) > \delta$ .

*Remark 2 (On the parameter  $\delta$ ).* Jumping ahead, both our FV-NIZK constructions in Sect. 4 has  $(\delta, \epsilon)$ -verification equivalence with  $\delta = 0$ . It seems that the only way to achieve verification equivalence is if the parameter  $\delta$  is either exactly 0 (as in our case) or large, but nothing in between.

Next, we adapt the *unbounded simulation soundness (USS)* of NIZK to our FV-NIZK. Recall that USS for NIZK and DV-NIZK ensures that a PPT adversary cannot generate a valid proof for a fresh and false statement  $c \notin \mathcal{L}_\rho$ , even if it can obtain multiple simulated proofs for instances not necessarily in  $\mathcal{L}_\rho$  [16, 30]. For FV-NIZK, we also allow the adversary to obtain many secret keys  $sk_d$  with  $d$  of its choices. Moreover, we consider a *strong* USS by giving the adversary multiple chances to win, following [16].

**Definition 6 (Strong USS).** *A tag-based FV-NIZK  $\Pi$  has strong USS, if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{uss}}(\lambda) := \Pr[\text{Exp}_{\Pi, \mathcal{A}}^{\text{uss}}(\lambda) \Rightarrow 1] \leq \text{negl}(\lambda),$$

where the experiment  $\text{Exp}_{\Pi, \mathcal{A}}^{\text{uss}}(\lambda)$  is defined in Fig. 2.

*Remark 3 (On the formalization of strong USS).* Note that in the strong USS experiment in Fig. 2,  $\text{SIM}(c, \tau)$  returns  $\perp$  directly if  $\tau$  was queried to  $\text{SIM}(\cdot, \cdot)$  before, following the definition of strong USS for DV-NIZK in [16]. Similar to [16], such a requirement is not an obstacle in many applications. For example, as we will see, in all our applications in Sect. 5,  $\tau$  is a hash of some random values. Thus  $\tau$  is different with overwhelming probability each time  $\text{SIM}(\cdot, \cdot)$  is invoked when the security of applications is reduced to the strong USS.

Moreover, we note that in the strong USS defined in [16],  $\text{VER}(\cdot, \tau, \cdot)$  also returns  $\perp$  if  $\tau$  was queried to  $\text{SIM}(\cdot, \tau)$  before, while ours does not have such a requirement. This relaxation seems reasonable when considering the security of NIZK, and it helps us to construct other cryptographic algorithms in a more straightforward way (e.g., constructing CCA-secure PKE without resorting to one-time signatures or authenticated encryption, as shown in Subsect. 5.2).

$\text{Exp}_{II, \mathcal{A}}^{uss}(\lambda):$ $\text{pp} \leftarrow \text{Par}(\mathbb{1}^\lambda, \mathcal{L}_\rho), (\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ $\mathcal{Q}_{sim} := \emptyset, \mathcal{Q}_{sk} := \emptyset$ $\text{win} := 0$ // A flag indicating whether $\mathcal{A}$ wins $\perp \leftarrow \mathcal{A}^{\text{SIM}(\cdot, \cdot), \text{DELEGATE}(\cdot), \text{VER}(\cdot, \cdot)}(\text{pp}, \text{crs})$ Output win  $\text{DELEGATE}(d):$ $sk_d \leftarrow \text{Delegate}(\text{msk}, d)$ $\mathcal{Q}_{sk} := \mathcal{Q}_{sk} \cup \{(d, sk_d)\}$ Return $sk_d$	$\text{SIM}(c, \tau):$ If $(\cdot, \tau, \cdot) \in \mathcal{Q}_{sim}$ : return $\perp$ $\pi \leftarrow \text{Sim}(\text{td}, c, \tau)$ $\mathcal{Q}_{sim} := \mathcal{Q}_{sim} \cup \{(c, \tau, \pi)\}$ Return $\pi$  $\text{VER}(c, \tau, \pi):$ If $(c, \tau, \pi) \in \mathcal{Q}_{sim}$ : return $\perp$ If $\text{MVer}(\text{msk}, c, \tau, \pi) = 1 \wedge c \notin \mathcal{L}_\rho$ : $\text{win} := 1$ Return $\text{MVer}(\text{msk}, c, \tau, \pi)$
--	--

**Fig. 2.** The strong USS experiment  $\text{Exp}_{II, \mathcal{A}}^{uss}(\lambda)$  for tag-based FV-NIZK.

Finally, we define *proof pseudorandomness* for FV-NIZK, which stipulates the pseudorandomness of proofs for PPT adversaries that are not given any secret key but allowed to access the verification oracle. Jumping ahead, this property serves as the core technical tool for the ciphertext pseudorandomness (thus anonymity) of our fine-grained verifiable PKE in Subsect. 5.2.

**Definition 7 (Proof Pseudorandomness).** *A tag-based FV-NIZK  $\Pi$  has proof pseudorandomness, if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{II, \mathcal{A}}^{pp}(\lambda) := |\Pr[\text{Exp}_{II, \mathcal{A}, 0}^{pp}(\lambda) \Rightarrow 1] - \Pr[\text{Exp}_{II, \mathcal{A}, 1}^{pp}(\lambda) \Rightarrow 1]| \leq \text{negl}(\lambda),$$

where the experiments  $\text{Exp}_{II, \mathcal{A}, \beta}^{pp}(\lambda)$  ( $\beta \in \{0, 1\}$ ) are defined in Fig. 3.

$\text{Exp}_{II, \mathcal{A}, \beta}^{pp}(\lambda):$ // $\beta \in \{0, 1\}$ $\text{pp} \leftarrow \text{Par}(\mathbb{1}^\lambda, \mathcal{L}_\rho), (\text{crs}, \text{td}, \text{msk}) \leftarrow \text{Gen}(\text{pp})$ $\mathcal{Q}_c := \emptyset, \mathcal{Q}_{sim} := \emptyset$ $\beta' \leftarrow \mathcal{A}^{\text{SAM}(\cdot), \text{SIM}(\cdot, \cdot), \text{VER}(\cdot, \cdot)}(\text{pp}, \text{crs})$ Output $\beta'$  $\text{VER}(c, \tau, \pi):$ If $(c, \tau, \pi) \in \mathcal{Q}_{sim}$ : return $\perp$ Return $\text{MVer}(\text{msk}, c, \tau, \pi)$	$\text{SAM}(\cdot):$ If $\beta = 0$ : $c \xleftarrow{\$} \mathcal{L}_\rho$ If $\beta = 1$ : $c \xleftarrow{\$} \mathcal{X}$ $\mathcal{Q}_c := \mathcal{Q}_c \cup \{c\}$ Return $c$	$\text{SIM}(c, \tau):$ If $c \notin \mathcal{Q}_c$ : return $\perp$ If $(\cdot, \tau, \cdot) \in \mathcal{Q}_{sim}$ : return $\perp$ If $\beta = 0$ : $\pi \leftarrow \text{Sim}(\text{td}, c, \tau)$ If $\beta = 1$ : $\pi \xleftarrow{\$} \mathcal{P}$ $\mathcal{Q}_c := \mathcal{Q}_c \setminus \{c\}$ $\mathcal{Q}_{sim} := \mathcal{Q}_{sim} \cup \{(c, \tau, \pi)\}$ Return $\pi$
---	--	--

**Fig. 3.** The proof pseudorandomness experiments  $\text{Exp}_{II, \mathcal{A}, \beta}^{pp}(\lambda)$  for tag-based FV-NIZK, where  $\mathcal{X}$  denotes the instance space, and  $\mathcal{P}$  denotes the proof space of  $\Pi$ .

*Remark 4 (On the formalization of proof pseudorandomness).* In fact, the proof pseudorandomness asks the pseudorandomness of proofs for instances *uniformly sampled* from the language  $\mathcal{L}_\rho$ . Moreover, the adversary  $\mathcal{A}$  in Fig. 3 has access to two oracles,  $\text{SAM}(\cdot)$  and  $\text{SIM}(\cdot, \cdot)$ , to obtain instances and simulated proofs,

respectively. In particular, the oracle  $\text{SIM}(c, \tau)$  returns proofs only for instances  $c$  output by  $\text{SAM}(\cdot)$ , but  $\tau$  can be determined by  $\mathcal{A}$ . Indeed, in certain applications of tag-based NIZK, the tag  $\tau$  may depend on the instance  $c$ . For example, in our application in PKE (cf. Subsect. 5.2),  $\tau$  is a hash of  $c$ . Our formalization captures such dependency between  $c$  and  $\tau$ .

*Remark 5 (Extension to the multi-user setting).* We can naturally extend the definitions of strong USS and proof pseudorandomness (i.e., Definitions 6 and 7) to the multi-user setting, and define strong  $\mu$ -USS and  $\mu$ -proof pseudorandomness in the setting of  $\mu \in \mathbb{N}$  users. The formal definitions can be found in the full version. More precisely, all  $\mu$  users share the same  $\text{pp}$  and each user  $i \in [\mu]$  invokes  $\text{Gen}(\text{pp})$  independently to get its own  $(\text{crs}^{(i)}, \text{td}^{(i)}, \text{msk}^{(i)})$ . Accordingly, the adversary  $\mathcal{A}$  has access to  $\text{SIM}(i, \cdot, \cdot)$ ,  $\text{DELEGATE}(i, \cdot)$ ,  $\text{VER}(i, \cdot, \cdot, \cdot)$  which additionally take a user index  $i \in [\mu]$  as input and prepare the responses using  $(\text{crs}^{(i)}, \text{td}^{(i)}, \text{msk}^{(i)})$ .

Jumping ahead, both the two schemes in Sect. 4 have almost tight strong USS (and the first one also have almost tight proof pseudorandomness) in the multi-user setting.

## 4 FV-NIZK for Linear Subspace Languages

In this section, we propose two tightly secure FV-NIZK schemes for linear subspace languages, based on the MDDH assumption. The first scheme is pairing-free and the second one relies on pairings.

Let  $\mathcal{G} = (\mathbb{G}, g, q)$  be a cyclic group  $\mathbb{G}$  of order  $q$  with generator  $g$ . Let  $\mathbf{A} \in \mathbb{Z}_q^{n_1 \times n_2}$  with  $n_1 > n_2$ . The linear subspace language is  $\mathcal{L}_{[\mathbf{A}]} := \text{Span}([\mathbf{A}]) := \{[\mathbf{c}] \mid \exists \mathbf{s} \in \mathbb{Z}_q^{n_2} \text{ s.t. } \mathbf{c} = \mathbf{A}\mathbf{s}\}$  with  $\mathbf{A}$  the trapdoor of  $\mathcal{L}_{[\mathbf{A}]}$ .

### 4.1 The First Construction without Pairings

Let  $m, k, n_1, n_2 \in \mathbb{N}$  and  $\mathcal{D}_{3k, k}$  be a matrix distribution. Let  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be a family of collision resistant hash functions. Our first construction of tag-based FV-NIZK  $\Pi$  is shown in Fig. 4, where the tag space is  $\mathcal{T} = \{0, 1\}^\lambda$  and the delegation space is  $\mathcal{D} = \mathbb{Z}_q^m$ . Note that this construction is pairing-free.

Completeness and perfect zero-knowledge follow directly from the fact that

$$\begin{aligned} \mathbf{u} &= (\mathbf{K}_0 + \theta\mathbf{K}_1)\mathbf{A}\mathbf{s} + \widehat{\mathbf{K}}_\tau\mathbf{B}\mathbf{r} = (\mathbf{K}_0 + \theta\mathbf{K}_1)\mathbf{c} + \widehat{\mathbf{K}}_\tau\mathbf{t} \quad // \text{ completeness (1)} \\ &= (\mathbf{K}_0 + \theta\mathbf{K}_1)\mathbf{c} + \widehat{\mathbf{K}}_\tau\mathbf{B}\mathbf{r}, \quad // \text{ perfect zero-knowledge} \end{aligned}$$

which implies  $\mathbf{d}^\top \mathbf{M}\mathbf{u} = \mathbf{d}^\top \mathbf{M}(\mathbf{K}_0 + \theta\mathbf{K}_1)\mathbf{c} + \mathbf{d}^\top \mathbf{M}\widehat{\mathbf{K}}_\tau\mathbf{t}$ . // completeness (2)

Next, we show the verification equivalence of  $\Pi$ .

**Theorem 1 (Verification Equivalence).** *The tag-based FV-NIZK scheme  $\Pi$  in Fig. 4 has  $(0, 1/q)$ -verification equivalence.*

$\text{Par}(1^\lambda, [\mathbf{A}] \in \mathbb{G}^{n_1 \times n_2}):$ $\mathbf{B} \leftarrow \mathcal{D}_{3k,k}; H \xleftarrow{\$} \mathcal{H}$ Return $\text{pp} := ([\mathbf{A}], [\mathbf{B}], H)$	$\text{MVer}(msk, [\mathbf{c}], \tau, \pi = ([\mathbf{t}], [\mathbf{u}])):$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}]); \widehat{\mathbf{K}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{K}}_{\ell, \tau_\ell}$ If $[\mathbf{u}] = (\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \widehat{\mathbf{K}}_\tau[\mathbf{t}]$ : return 1 Otherwise: return 0
$\text{Gen}(\text{pp}):$ $\mathbf{K}_0, \mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_q^{(m+1) \times n_1}; \mathbf{M} \xleftarrow{\$} \mathbb{Z}_q^{m \times (m+1)}$ For $\ell \in [\lambda], b \in \{0, 1\}$ : $\widehat{\mathbf{K}}_{\ell, b} \xleftarrow{\$} \mathbb{Z}_q^{(m+1) \times 3k}$ $\text{crs} := ([\mathbf{K}_0 \mathbf{A}], [\mathbf{K}_1 \mathbf{A}], \{[\widehat{\mathbf{K}}_{\ell, b} \mathbf{B}]\}_{\ell, b})$ $\text{td} := (\mathbf{K}_0, \mathbf{K}_1)$ $msk := (\mathbf{K}_0, \mathbf{K}_1, \{\widehat{\mathbf{K}}_{\ell, b}\}_{\ell, b}, \mathbf{M})$ Return $(\text{crs}, \text{td}, msk)$	$\text{Sim}(\text{td}, [\mathbf{c}], \tau):$ $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k; [\mathbf{t}] := [\mathbf{B}]\mathbf{r}$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}]); \widehat{\mathbf{K}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{K}}_{\ell, \tau_\ell}$ $[\mathbf{u}] := (\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + [\widehat{\mathbf{K}}_\tau \mathbf{B}]\mathbf{r} \in \mathbb{G}^{m+1}$ Return $\pi := ([\mathbf{t}], [\mathbf{u}])$
$\text{Prove}(\text{crs}, [\mathbf{c}], \text{s}, \tau): // \mathbf{c} = \text{As}$ $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k; [\mathbf{t}] := [\mathbf{B}]\mathbf{r}$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}]); \widehat{\mathbf{K}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{K}}_{\ell, \tau_\ell}$ $[\mathbf{u}] := [(\mathbf{K}_0 + \theta \mathbf{K}_1)\mathbf{A}]\text{s} + [\widehat{\mathbf{K}}_\tau \mathbf{B}]\mathbf{r} \in \mathbb{G}^{m+1}$ Return $\pi := ([\mathbf{t}], [\mathbf{u}])$	$\text{Delegate}(msk, \mathbf{d} \in \mathbb{Z}_q^m):$ Return $sk_{\mathbf{d}} := (\mathbf{d}^\top \mathbf{M}, \mathbf{d}^\top \mathbf{M} \mathbf{K}_0, \mathbf{d}^\top \mathbf{M} \mathbf{K}_1, \{\mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}}_{\ell, b}\}_{\ell, b})$
$\text{FVer}(sk_{\mathbf{d}}, [\mathbf{c}], \tau, \pi = ([\mathbf{t}], [\mathbf{u}])):$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}]); \widehat{\mathbf{K}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{K}}_{\ell, \tau_\ell}$ If $\mathbf{d}^\top \mathbf{M}[\mathbf{u}] = \mathbf{d}^\top \mathbf{M}(\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}}_\tau[\mathbf{t}]$ : return 1 Otherwise: return 0	$\text{FVer}(sk_{\mathbf{d}}, [\mathbf{c}], \tau, \pi = ([\mathbf{t}], [\mathbf{u}])):$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}]); \widehat{\mathbf{K}}_\tau := \sum_{\ell=1}^\lambda \widehat{\mathbf{K}}_{\ell, \tau_\ell}$ If $\mathbf{d}^\top \mathbf{M}[\mathbf{u}] = \mathbf{d}^\top \mathbf{M}(\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}}_\tau[\mathbf{t}]$ : return 1 Otherwise: return 0

**Fig. 4.** The pairing-free construction of tag-based FV-NIZK II.

*Proof.* The first property ( $\text{MVer} \implies \text{FVer}$ ) is straightforward, since  $[\mathbf{u}] = (\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \widehat{\mathbf{K}}_\tau[\mathbf{t}]$  directly implies  $\mathbf{d}^\top \mathbf{M}[\mathbf{u}] = \mathbf{d}^\top \mathbf{M}(\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}}_\tau[\mathbf{t}]$ .

To show the second property ( $\text{MVer} \xleftarrow{w.h.p.} \text{FVer}$ ), we consider an (unbounded) adversary  $\mathcal{A}$  that finally outputs  $([\mathbf{c}^*], \tau^*, \pi^* = ([\mathbf{t}^*], [\mathbf{u}^*]), \mathbf{d}^*)$  in the experiment  $\text{Exp}_{II, \mathcal{A}, 0}^{\text{ver-equ}}(\lambda)$  (cf. Fig. 1). Let  $\mathbf{D}$  denote the matrix consisting of all vectors  $\mathbf{d}$  that  $\mathcal{A}$  queried  $\text{DELEGATE}(\cdot)$ . We analyze  $\mathcal{A}$ 's advantage as follows.

Note that the algorithm  $\text{Delegate}$  is deterministic and linear in  $\mathbf{d}$ . That is, if  $\mathbf{d}^* \in \text{Span}(\mathbf{D})$ , then  $sk_{\mathbf{d}^*}$  is totally determined by  $\mathcal{Q}_{sk} = \{(\mathbf{d}, sk_{\mathbf{d}})\}$  and  $\mathbf{d}^*$ , and hence has no entropy left at all. Therefore, for  $\mathcal{A}$  to win,  $\widetilde{\mathbf{H}}_\infty(sk_{\mathbf{d}^*} | \text{crs}, \mathcal{Q}_{sim}, \mathcal{Q}_{sk}, \mathbf{d}^*) > 0$  holds, and we must have  $\mathbf{d}^* \notin \text{Span}(\mathbf{D})$ . Moreover, since the algorithm  $\text{Sim}$  does not involve  $\mathbf{M}$  at all,  $\mathcal{A}$  obtains nothing about  $\mathbf{M}$  from  $\text{SIM}(\cdot, \cdot)$ . Thus,  $\mathbf{d}^* \notin \text{Span}(\mathbf{D})$  implies that  $\mathbf{d}^{*\top} \mathbf{M}$  is uniformly random over  $\mathbb{Z}_q^{1 \times (m+1)}$  from  $\mathcal{A}$ 's view. And consequently, the event  $\text{FVer}(sk_{\mathbf{d}^*}, [\mathbf{c}^*], \tau^*, \pi^*) = 1 \wedge \text{MVer}(msk, [\mathbf{c}^*], \tau^*, \pi^*) = 0$ , i.e.,

$$\mathbf{d}^{*\top} \mathbf{M} \underbrace{\left( \mathbf{u}^* - (\mathbf{K}_0 + \theta^* \mathbf{K}_1) \mathbf{c}^* - \widehat{\mathbf{K}}_{\tau^*} \mathbf{t}^* \right)}_{\neq \mathbf{0}} = 0,$$

occurs with probability at most  $1/q$ . This shows  $\text{Adv}_{II, \mathcal{A}, 0}^{\text{ver-equ}}(\lambda) \leq 1/q$ .  $\square$

Now we show that  $II$  has almost tight strong USS and almost tight proof pseudorandomness via the following two theorems.

**Theorem 2 (Almost Tight Strong USS).** *If the  $\mathcal{D}_{3k,k}$ -MDDH assumption holds in  $\mathbb{G}$  and  $\mathcal{H}$  is a family of collision resistant hash functions, then the tag-based FV-NIZK scheme  $II$  in Fig. 4 has strong USS. More precisely, for any*

adversary  $\mathcal{A}$  against the strong USS security of  $\Pi$ , there exist algorithms  $\mathcal{B}_1, \mathcal{B}_2$  s.t.  $\max(\text{Time}(\mathcal{B}_1), \text{Time}(\mathcal{B}_2)) \approx \text{Time}(\mathcal{A}) + (Q_{sim} + Q_{ver} + Q_{del}) \cdot \text{poly}(\lambda)$ , and

$$\text{Adv}_{\Pi, \mathcal{A}}^{uss}(\lambda) \leq \text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{cr}(\lambda) + (8\lambda k + 2k) \cdot \text{Adv}_{\mathcal{D}_{3k, k}, \mathbb{G}, \mathcal{B}_2}^{mddh}(\lambda) + \frac{(2\lambda+2)Q_{ver}+4\lambda+1}{q-1},$$

where  $Q_{sim}, Q_{ver}, Q_{del}$  denote the numbers of queries to SIM, VER, DELEGATE, respectively.

**Theorem 3 (Almost Tight Proof Pseudorandomness).** *Let  $n_1 \geq 2n_2$ . If the  $\mathcal{D}_{n_1, n_2}$ -MDDH assumption and the  $\mathcal{D}_{3k, k}$ -MDDH assumption hold in  $\mathbb{G}$ , and  $\mathcal{H}$  is a family of collision resistant hash functions, then the tag-based FV-NIZK scheme  $\Pi$  in Fig. 4 has proof pseudorandomness. More precisely, for any adversary  $\mathcal{A}$  against the proof pseudorandomness of  $\Pi$ , there exist algorithms  $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$  s.t.  $\max(\text{Time}(\mathcal{B}_1), \text{Time}(\mathcal{B}_2), \text{Time}(\mathcal{B}_3)) \approx \text{Time}(\mathcal{A}) + (Q_{sim} + Q_{ver}) \cdot \text{poly}(\lambda)$ , and*

$$\begin{aligned} \text{Adv}_{\Pi, \mathcal{A}}^{pp}(\lambda) \leq & (n_1 - n_2 + 2) \text{Adv}_{\mathcal{D}_{n_1, n_2}, \mathbb{G}, \mathcal{B}_1}^{mddh}(\lambda) + (16\lambda k + 6k) \text{Adv}_{\mathcal{D}_{3k, k}, \mathbb{G}, \mathcal{B}_2}^{mddh}(\lambda) \\ & + 2 \text{Adv}_{\mathcal{H}, \mathcal{B}_3}^{cr}(\lambda) + \frac{(4\lambda+4)Q_{ver}+8\lambda+6}{q-1}, \end{aligned}$$

where  $Q_{sim}$  and  $Q_{ver}$  denote the numbers of queries to SIM and VER, respectively.

We prove Theorems 2 and 3 in our full version due to space limitations. See Sect. 1 for a high-level proof sketch.

*Remark 6 (On the almost tightness of strong USS and proof pseudorandomness).* The terms  $\frac{(2\lambda+2)Q_{ver}+4\lambda+1}{q-1}$  and  $\frac{(4\lambda+4)Q_{ver}+8\lambda+6}{q-1}$  in Theorem 2 and Theorem 3 do not affect the tightness of the reductions since they are statistically small. Moreover,  $n_1, n_2, k$  are parameters of the MDDH assumptions and are constants (e.g.,  $n_1 = 2, n_2 = 1, k = 1$ ). Consequently, the strong USS and proof pseudorandomness have security loss factors  $O(\lambda)$ , and thus are almost tight.

## 4.2 The Second Construction with Pairings

Let  $m, k, n_1, n_2 \in \mathbb{N}$  and  $\mathcal{D}_{2k, k}$  be a matrix distribution. Let  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be a family of collision resistant hash functions. Similar to [4], we use a NIZK proof  $\Pi_{or} = (\Pi_{or}.Gen, \Pi_{or}.TGen, \Pi_{or}.Prove, \Pi_{or}.Sim, \Pi_{or}.Ver)$  for OR-language  $\mathcal{L}_{[\mathbf{B}_0], [\mathbf{B}_1]}^\vee := \text{Span}([\mathbf{B}_0]) \cup \text{Span}([\mathbf{B}_1]) := \{[\mathbf{t}] \mid \exists \mathbf{r} \in \mathbb{Z}_q^k \text{ s.t. } \mathbf{t} = \mathbf{B}_0 \mathbf{r} \vee \mathbf{t} = \mathbf{B}_1 \mathbf{r}\}$  as a building block, where  $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_q^{2k \times k}$ . We refer our full version for the syntax of NIZK proofs and a concrete MDDH-based scheme of  $\Pi_{or}$  proposed in [18, 29]. Our second construction of tag-based FV-NIZK  $\Pi$  is shown in Fig. 5, where the tag space is  $\mathcal{T} = \{0, 1\}^*$  and the delegation space is  $\mathcal{D} = \mathbb{Z}_q^m$ . Note that compared to the QA-NIZK scheme proposed in [4], our FV-NIZK scheme uses less pairing operations, since only  $\Pi_{or}.Ver$  involves pairings.

$\text{Par}(1^\lambda, [\mathbf{A}] \in \mathbb{G}^{n_1 \times n_2});$ $\mathbf{B}_0, \mathbf{B}_1 \leftarrow \mathcal{D}_{2k,k}; H \xleftarrow{\$} \mathcal{H}$ $\text{crs}_{or} \leftarrow \Pi_{or}.\text{Gen}(1^\lambda, [\mathbf{B}_0], [\mathbf{B}_1])$ Return $\text{pp} := ([\mathbf{A}], [\mathbf{B}_0], \text{crs}_{or}, H)$	$\text{MVer}(msk, [\mathbf{c}], \tau, \pi = ([\mathbf{t}], [\mathbf{u}], \pi_{or})):$ If $\Pi_{or}.\text{Ver}(\text{crs}_{or}, [\mathbf{t}], \pi_{or}) = 0$ : return 0 $\theta := H([\mathbf{c}], \tau, [\mathbf{t}], \pi_{or})$ If $[\mathbf{u}] = (\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \widehat{\mathbf{K}}[\mathbf{t}]$ : return 1 Otherwise: return 0
$\text{Gen}(\text{pp}):$ $\mathbf{K}_0, \mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_q^{(m+1) \times n_1}$ $\widehat{\mathbf{K}} \xleftarrow{\$} \mathbb{Z}_q^{(m+1) \times 2k}; \mathbf{M} \xleftarrow{\$} \mathbb{Z}_q^{m \times (m+1)}$ $\text{crs} := ([\mathbf{K}_0 \mathbf{A}], [\mathbf{K}_1 \mathbf{A}], [\widehat{\mathbf{K}} \mathbf{B}_0])$ $\text{td} := (\mathbf{K}_0, \mathbf{K}_1)$ $msk := (\mathbf{K}_0, \mathbf{K}_1, \widehat{\mathbf{K}}, \mathbf{M})$ Return $(\text{crs}, \text{td}, msk)$	$\text{Sim}(\text{td}, [\mathbf{c}], \tau):$ $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k; [\mathbf{t}] := [\mathbf{B}_0] \mathbf{r}$ $\pi_{or} \leftarrow \Pi_{or}.\text{Prove}(\text{crs}_{or}, [\mathbf{t}], \mathbf{r})$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}], \pi_{or})$ $[\mathbf{u}] := (\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + [\widehat{\mathbf{K}} \mathbf{B}_0] \mathbf{r} \in \mathbb{G}^{m+1}$ Return $\pi := ([\mathbf{t}], [\mathbf{u}], \pi_{or})$
$\text{Prove}(\text{crs}, [\mathbf{c}], \mathbf{s}, \tau): // \mathbf{c} = \mathbf{A} \mathbf{s}$ $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k; [\mathbf{t}] := [\mathbf{B}_0] \mathbf{r}$ $\pi_{or} \leftarrow \Pi_{or}.\text{Prove}(\text{crs}_{or}, [\mathbf{t}], \mathbf{r})$ $\theta := H([\mathbf{c}], \tau, [\mathbf{t}], \pi_{or})$ $[\mathbf{u}] := [(\mathbf{K}_0 + \theta \mathbf{K}_1) \mathbf{A}] \mathbf{s} + [\widehat{\mathbf{K}} \mathbf{B}_0] \mathbf{r} \in \mathbb{G}^{m+1}$ Return $\pi := ([\mathbf{t}], [\mathbf{u}], \pi_{or})$	$\text{Delegate}(msk, \mathbf{d} \in \mathbb{Z}_q^m):$ Return $sk_{\mathbf{d}} := (\mathbf{d}^\top \mathbf{M}, \mathbf{d}^\top \mathbf{M} \mathbf{K}_0, \mathbf{d}^\top \mathbf{M} \mathbf{K}_1, \mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}})$
	$\text{FVer}(sk_{\mathbf{d}}, [\mathbf{c}], \tau, \pi = ([\mathbf{t}], [\mathbf{u}], \pi_{or})):$ If $\Pi_{or}.\text{Ver}(\text{crs}_{or}, [\mathbf{t}], \pi_{or}) = 0$ : return 0 $\theta := H([\mathbf{c}], \tau, [\mathbf{t}], \pi_{or})$ If $\mathbf{d}^\top \mathbf{M}[\mathbf{u}] = \mathbf{d}^\top \mathbf{M}(\mathbf{K}_0 + \theta \mathbf{K}_1)[\mathbf{c}] + \mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}}[\mathbf{t}]$ : return 1 Otherwise: return 0

**Fig. 5.** The pairing-based construction of tag-based FV-NIZK  $\Pi$ , where  $\Pi_{or} = (\Pi_{or}.\text{Gen}, \Pi_{or}.\text{TGen}, \Pi_{or}.\text{Prove}, \Pi_{or}.\text{Sim}, \Pi_{or}.\text{Ver})$  is a NIZK proof for OR-language  $\mathcal{L}_{[\mathbf{B}_0], [\mathbf{B}_1]}^\vee$ .

Completeness and perfect zero-knowledge follow directly from the fact that

$$\begin{aligned} \mathbf{u} &= (\mathbf{K}_0 + \theta \mathbf{K}_1) \mathbf{A} \mathbf{s} + \widehat{\mathbf{K}} \mathbf{B}_0 \mathbf{r} = (\mathbf{K}_0 + \theta \mathbf{K}_1) \mathbf{c} + \widehat{\mathbf{K}} \mathbf{t} \quad // \text{ completeness (1)} \\ &= (\mathbf{K}_0 + \theta \mathbf{K}_1) \mathbf{c} + \widehat{\mathbf{K}} \mathbf{B}_0 \mathbf{r}, \quad // \text{ perfect zero-knowledge} \end{aligned}$$

which implies  $\mathbf{d}^\top \mathbf{M} \mathbf{u} = \mathbf{d}^\top \mathbf{M}(\mathbf{K}_0 + \theta \mathbf{K}_1) \mathbf{c} + \mathbf{d}^\top \mathbf{M} \widehat{\mathbf{K}} \mathbf{t}$ . // completeness (2)

Next, we show the verification equivalence and almost tight strong USS of  $\Pi$ .

**Theorem 4 (Verification Equivalence).** *The tag-based FV-NIZK scheme  $\Pi$  in Fig. 5 has  $(0, 1/q)$ -verification equivalence.*

The proof is very similar to that of Theorem 1 and we show it in the full version.

**Theorem 5 (Almost Tight Strong USS).** *If the  $\mathcal{D}_{2k,k}$ -MDDH assumption holds in  $\mathbb{G}$ ,  $\mathcal{H}$  is a family of collision resistant hash functions, and  $\Pi_{or}$  is a NIZK proof for  $\mathcal{L}_{[\mathbf{B}_0], [\mathbf{B}_1]}^\vee$  with completeness, perfect soundness and zero-knowledge, then the tag-based FV-NIZK scheme  $\Pi$  in Fig. 5 has strong USS. More precisely, for any adversary  $\mathcal{A}$  against the strong USS security of  $\Pi$ , there exist algorithms  $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$  s.t.  $\max(\text{Time}(\mathcal{B}_1), \text{Time}(\mathcal{B}_2), \text{Time}(\mathcal{B}_3)) \approx \text{Time}(\mathcal{A}) + (Q_{sim} + Q_{ver} + Q_{del}) \cdot \text{poly}(\lambda)$ , and*



$$\begin{aligned} \text{Adv}_{\Pi, \mathcal{A}}^{\text{uss}}(\lambda) \leq & \text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + (2n + 2) \cdot \text{Adv}_{\Pi_{or}, \mathcal{B}_2}^{zk}(\lambda) \\ & + (4kn + 2k) \cdot \text{Adv}_{\mathcal{D}_{2k, k, \mathbb{G}}, \mathcal{B}_3}^{\text{mddh}}(\lambda) + \frac{(n+1)(Q_{sim}Q_{ver}+4)}{q-1}. \end{aligned}$$

where  $Q_{sim}, Q_{ver}, Q_{del}$  denote the numbers of queries to SIM, VER, DELEGATE, respectively, and  $n := \lceil \log Q_{sim} \rceil$ .

The proof is provided in the full version due to space limitations.

*Remark 7 (On the almost tightness of strong USS).* Similar to Remark 6, the term  $\frac{(n+1)(Q_{sim}Q_{ver}+4)}{q-1}$  in Theorem 5 does not affect the tightness of the reduction since it is statistically small. Moreover,  $k$  is the parameter of the MDDH assumption (e.g.,  $k = 1$  corresponds to the standard DDH assumption). Consequently, the strong USS has security loss factor  $O(n) = O(\lceil \log Q_{sim} \rceil)$ , which is  $O(\log \lambda)$  for PPT adversaries due to  $Q_{sim} = \text{poly}(\lambda)$ , and thus is almost tight.

*Remark 8* We note that our tag-based FV-NIZK scheme  $\Pi$  in Fig. 5 does not achieve proof pseudorandomness, since its proof  $\pi$  contains a proof  $\pi_{or}$  of the underlying NIZK scheme  $\Pi_{or}$  which supports public verification, so that anyone who obtains  $\text{crs}_{or}$  from  $\text{pp}$  can check the validity of  $\pi_{or}$ .

## 5 Applications of FV-NIZK

In this section, we illustrate the usefulness of tag-based FV-NIZK by showing two applications, including CCA-secure IPFE in Subsect. 5.1 and CCA-secure fine-grained verifiable PKE (FV-PKE) in Subsect. 5.2.

By instantiating with the almost tightly secure FV-NIZK schemes constructed in Sect. 4, we immediately obtain IPFE and FV-PKE schemes that achieve almost tight mCCA (multi-challenge CCA) security. Moreover, the resulting schemes are either pairing-free (when using the FV-NIZK scheme in Subsect. 4.1), or use less pairing operations than existing works (when using the FV-NIZK scheme in Subsect. 4.2).

### 5.1 Almost Tightly mCCA-Secure IPFE Schemes

In [26], Liu et al. proposed the first almost tightly mCCA secure IPFE scheme, based on a tightly mCPA secure scheme [31] and an almost tightly secure QA-NIZK argument for linear subspace languages [4]. However, the QA-NIZK argument in [4] involves pairings, so does Liu et al.'s IPFE.

To reduce the number of pairing operations or even get rid of pairings, we replace the QA-NIZK with our tag-based FV-NIZK for linear subspace languages in the IPFE construction. When the tag-based FV-NIZK is instantiated with the construction in Subsect. 4.1, we obtain the first pairing-free IPFE scheme with almost tight mCCA security. When it is instantiated with the construction in Subsect. 4.2, we obtain a pairing-based IPFE scheme that uses less pairing operations than [26].

Formally, we present the syntax of IPFE and its mCCA security in the full version and describe our IPFE construction as follows. Let  $m, k, X, Y \in \mathbb{N}$ , and let  $\mathcal{D}_k$  be a matrix distribution. Let  $\Pi = (\Pi.\text{Par}, \Pi.\text{Gen}, \Pi.\text{Prove}, \Pi.\text{MVer}, \Pi.\text{Sim}, \Pi.\text{Delegate}, \Pi.\text{FVer})$  be a tag-based FV-NIZK for linear subspace language  $\mathcal{L}_{[\mathbf{A}]}$  with tag space  $\mathcal{T}$  and delegation space  $\mathcal{D} = \mathbb{Z}_q^m$ . Let  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathcal{T}$  be a family of collision resistant hash functions. Our IPFE construction  $\text{IPFE}_{\text{mcca}} = (\text{Par}, \text{Setup}, \text{Enc}, \text{KeyGen}, \text{Dec})$  is described in Fig. 6, where the message space is  $[-X, X]^m \subseteq \mathbb{Z}_q^m$  and the inner product function is defined by  $\mathbf{y} \in [-Y, Y]^m \subseteq \mathbb{Z}_q^m$ . Similar to [26, 31], we require  $mXY$  to be a polynomial in  $\lambda$ .

The correctness of  $\text{IPFE}_{\text{mcca}}$  follows from the completeness of  $\Pi$  and the fact that for  $\mathbf{x} \in [-X, X]^m$  and  $\mathbf{y} \in [-Y, Y]^m$ , it holds

$$d = \mathbf{y}^\top (\mathbf{W}\mathbf{A}\mathbf{s} + \mathbf{x}) - \mathbf{y}^\top \mathbf{W}(\mathbf{A}\mathbf{s}) = \mathbf{y}^\top \mathbf{x} \in [-mXY, mXY].$$

$\text{Par}(1^\lambda):$ $\tilde{\mathbf{A}} \leftarrow \mathcal{D}_k; \mathbf{A} := \mathbf{I}_{km} \otimes \tilde{\mathbf{A}}$ $\widehat{\text{pp}} \leftarrow \Pi.\text{Par}(1^\lambda, [\mathbf{A}]), H \xleftarrow{\$} \mathcal{H}$ Return $\text{pp} := ([\tilde{\mathbf{A}}], \widehat{\text{pp}}, H)$ $\text{Setup}(1^m, \text{pp}):$ $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{m \times k(k+1)m}$ $(\text{crs}, \text{td}, \widehat{msk}) \leftarrow \Pi.\text{Gen}(\Pi.\text{pp})$ Return $\text{mpk} := ([\mathbf{W}\mathbf{A}], \text{crs}), \text{msk} := (\mathbf{W}, \widehat{msk})$ $\text{KeyGen}(\text{msk}, \mathbf{y} \in [-Y, Y]^m):$ $\widehat{sk}_{\mathbf{y}} \leftarrow \Pi.\text{Delegate}(\widehat{msk}, \mathbf{y})$ Return $sk_{\mathbf{y}} := (\mathbf{y}, \mathbf{y}^\top \mathbf{W}, \widehat{sk}_{\mathbf{y}})$	$\text{Enc}(\text{mpk}, \mathbf{x} \in [-X, X]^m):$ $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{k^2 m}; [\mathbf{c}] := [\mathbf{A}]\mathbf{s} \in \mathbb{G}^{k(k+1)m}$ $[\mathbf{v}] := [\mathbf{W}\mathbf{A}]\mathbf{s} + [\mathbf{x}] \in \mathbb{Z}_q^m$ $\tau := H(\text{mpk}, [\mathbf{c}], [\mathbf{v}])$ $\pi \leftarrow \Pi.\text{Prove}(\text{crs}, [\mathbf{c}], \mathbf{s}, \tau)$ Return $ct := ([\mathbf{c}], [\mathbf{v}], \pi)$ $\text{Dec}(sk_{\mathbf{y}}, ct):$ Parse $ct = ([\mathbf{c}], [\mathbf{v}], \pi)$ $\tau := H(\text{mpk}, [\mathbf{c}], [\mathbf{v}])$ If $\Pi.\text{FVer}(\widehat{sk}_{\mathbf{y}}, [\mathbf{c}], \tau, \pi) = 1$ : $[d] := \mathbf{y}^\top [\mathbf{v}] - \mathbf{y}^\top \mathbf{W}[\mathbf{c}]$ Return $d \in [-mXY, mXY]$ Otherwise: return $\perp$
--	---

**Fig. 6.** Construction of  $\text{IPFE}_{\text{mcca}}$  from tag-based FV-NIZK  $\Pi$ . For the ease of reading, we emphasize different parts with [26] in gray boxes.

**Theorem 6 (Almost Tight mCCA Security of  $\text{IPFE}_{\text{mcca}}$ ).** *If the  $\mathcal{D}_k$ -MDDH assumption holds in  $\mathbb{G}$ ,  $\mathcal{H}$  is a family of collision resistant hash functions, and  $\Pi$  is a tag-based FV-NIZK with  $(0, \epsilon)$ -verification equivalence and strong USS, then  $\text{IPFE}_{\text{mcca}}$  shown in Fig. 6 is mCCA-secure. Concretely, for any PPT adversary  $\mathcal{A}$ , there exist PPT algorithms  $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$  s.t.  $\max(\text{Time}(\mathcal{B}_1), \text{Time}(\mathcal{B}_2), \text{Time}(\mathcal{B}_3)) \approx \text{Time}(\mathcal{A}) + (Q_{\text{enc}} + Q_{sk} + Q_{\text{dec}}) \cdot \text{poly}(\lambda, m)$  with  $\text{poly}(\lambda, m)$  independent of  $\mathcal{A}$ , and*

$$\text{Adv}_{\text{IPFE}_{\text{mcca}}, \mathcal{A}}^{\text{mcca}}(\lambda) \leq 2\text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + 4\text{Adv}_{\mathcal{D}_k, \mathbb{G}, \mathcal{B}_2}^{\text{mddh}}(\lambda) + 2\text{Adv}_{\Pi, \mathcal{B}_3}^{\text{uss}}(\lambda) + 2Q_{\text{dec}} \cdot \epsilon + \frac{2}{q-1},$$

where  $Q_{enc}$ ,  $Q_{sk}$  and  $Q_{dec}$  denote the total numbers of encryption, key generation and decryption queries, respectively.

The proof is shown in the full version due to space limitations.

## 5.2 Almost Tightly mCCA-Secure FV-PKE Schemes

In this subsection, we formalize the new primitive called *Fine-grained Verifiable PKE (FV-PKE)*, and define verification soundness, mCCA security, and ciphertext pseudorandomness for it. Then we show how to construct FV-PKE based on our tag-based FV-NIZK. By instantiating with the almost tightly secure FV-NIZK scheme proposed in Subsect. 4.1, we obtain the first FV-PKE scheme with almost tight mCCA security and ciphertext pseudorandomness.

We first present the syntax of FV-PKE.

**Definition 8 (FV-PKE).** A *Fine-grained Verifiable Public-Key Encryption (FV-PKE)* scheme consists of six PPT algorithms, namely  $\text{FPKE} = (\text{Par}, \text{Gen}, \text{Enc}, \text{Dec}, \text{Delegate}, \text{Ver})$ .

- $\text{pp} \leftarrow \text{Par}(1^\lambda)$ : Initialization algorithm takes the security parameter  $\lambda$  as input and outputs a public parameter  $\text{pp}$ , which defines the message space  $\mathcal{M}$  and the delegation space  $\mathcal{D}$ .
- $(pk, sk) \leftarrow \text{Gen}(\text{pp})$ : Generation algorithm takes  $\text{pp}$  as inputs, and outputs a public key  $pk$  and a secret key  $sk$ . We assume  $pk$  contains  $\text{pp}$ , and it serves as an implicit input of  $\text{Enc}$ ,  $\text{Dec}$ ,  $\text{Delegate}$ , and  $\text{Ver}$ .
- $ct \leftarrow \text{Enc}(pk, M)$ : Encryption algorithm takes  $pk$  and a message  $M \in \mathcal{M}$  as inputs, and outputs a ciphertext  $ct$ .
- $M'/\perp \leftarrow \text{Dec}(sk, ct)$ : Decryption algorithm takes  $sk$  and a ciphertext  $ct$  as inputs, and outputs a message  $M' \in \mathcal{M}$  or a special failure symbol  $\perp$ .
- $sk_d \leftarrow \text{Delegate}(sk, d)$ : Delegation algorithm takes  $sk$  and a delegation  $d \in \mathcal{D}$  as inputs, and outputs a delegated secret key  $sk_d$ .
- $0/1 \leftarrow \text{Ver}(sk_d, ct)$ : Verification algorithm takes  $sk_d$  and  $ct$  as inputs, and outputs a bit indicating whether  $ct$  is a valid ciphertext or not.

We require FPKE to have decryption correctness and verification correctness.

**Decryption Correctness.** For all  $\text{pp}$ ,  $(pk, sk) \leftarrow \text{Gen}(\text{pp})$ ,  $M \in \mathcal{M}$  and  $ct \leftarrow \text{Enc}(pk, M)$ , it holds that  $\text{Dec}(sk, ct) = M$ .

**Verification Correctness.** For all  $\text{pp}$ ,  $(pk, sk) \leftarrow \text{Gen}(\text{pp})$ ,  $M \in \mathcal{M}$  and  $ct \leftarrow \text{Enc}(pk, M)$ , it holds  $\text{Ver}(sk_d, ct) = 1$  for all  $sk_d \leftarrow \text{Delegate}(sk, d)$  of all  $d \in \mathcal{D}$ .

Note that the first four algorithms ( $\text{Par}$ ,  $\text{Gen}$ ,  $\text{Enc}$ ,  $\text{Dec}$ ) of FV-PKE basically constitute a standard PKE scheme. Moreover, the two additional algorithms ( $\text{Delegate}$ ,  $\text{Ver}$ ) provide the fine-grained ability for verifying ciphertext validity.

Next, we define a statistical property called *verification soundness* for FV-PKE. Loosely speaking, it essentially requires that for any ciphertext  $ct$  and any  $sk_d$ ,  $\text{Ver}(sk_d, ct)$  outputs 1 if and only if  $ct$  is a valid ciphertext, i.e.,  $\text{Dec}(sk, ct)$  succeeds, except for a negligible probability.

**Definition 9 (Verification Soundness of FV-PKE).** Let  $\delta, \epsilon > 0$ . An FV-PKE scheme FPKE has  $(\delta, \epsilon)$ -verification soundness, if for any (even unbounded) adversary  $\mathcal{A}$ , it holds that

$$\text{Adv}_{\text{FPKE}, \mathcal{A}, \delta}^{\text{ver-snd}}(\lambda) := \Pr[\text{Exp}_{\text{FPKE}, \mathcal{A}, \delta}^{\text{ver-snd}}(\lambda) \Rightarrow 1] \leq \epsilon,$$

where the experiment  $\text{Exp}_{\text{FPKE}, \mathcal{A}, \delta}^{\text{ver-snd}}(\lambda)$  is defined in Fig. 7.

$\text{Exp}_{\Pi, \mathcal{A}, \delta}^{\text{ver-snd}}(\lambda):$ $\text{pp} \leftarrow \text{Par}(1^\lambda), (pk, sk) \leftarrow \text{Gen}(\text{pp}), \mathcal{Q}_{sk} := \emptyset$ $(ct^*, d^*) \leftarrow \mathcal{A}^{\text{DELEGATE}(\cdot)}(\text{pp}, pk)$ $sk_{d^*} \leftarrow \text{Delegate}(sk, d^*)$ If $\tilde{\mathbf{H}}_\infty(sk_{d^*}   pk, \mathcal{Q}_{sk}, d^*) > \delta$ $\wedge \left( \begin{array}{l} (\text{Ver}(sk_{d^*}, ct^*) = 1 \wedge \text{Dec}(sk, ct^*) = \perp) \\ \vee (\text{Ver}(sk_{d^*}, ct^*) = 0 \wedge \text{Dec}(sk, ct^*) \neq \perp) \end{array} \right): \text{output } 1$ Otherwise: output 0	$\text{DELEGATE}(d):$ $sk_d \leftarrow \text{Delegate}(sk, d)$ $\mathcal{Q}_{sk} := \mathcal{Q}_{sk} \cup \{(d, sk_d)\}$ Return $sk_d$
---	--

**Fig. 7.** The verification soundness experiment  $\text{Exp}_{\text{FPKE}, \mathcal{A}, \delta}^{\text{ver-snd}}(\lambda)$  for FV-PKE.

*Remark 9 (On the formalization of verification soundness).* We stress that we do not require  $\text{Ver}$  can always correctly decide whether a ciphertext is valid or not. That is, there might exist a ciphertext  $ct$  and a pair  $(d, sk_d)$  s.t.,  $\text{Dec}(sk, ct) = \perp$  but  $\text{Ver}(sk_d, ct) = 1$ , or  $\text{Dec}(sk, ct) \neq \perp$  but  $\text{Ver}(sk_d, ct) = 0$ . Nevertheless, verification soundness of FV-PKE ensures that even for an (unbounded) adversary  $\mathcal{A}$ , if it does not get enough information about  $sk_{d^*}$  (and thus  $sk$ ), it is hard for  $\mathcal{A}$  to find a  $ct^*$  that makes  $\text{Dec}(sk, \cdot)$  and  $\text{Ver}(sk_{d^*}, \cdot)$  perform inconsistently. Similar to Remark 1, we require “ $\tilde{\mathbf{H}}_\infty(sk_{d^*} | pk, \mathcal{Q}_{sk}, d^*) > \delta$ ” in Fig. 7 to prevent trivial attacks, since for those who get  $sk_{d^*}$ , it might be easy for them to produce such a  $ct^*$ .

*Remark 10 (On the motivation for defining FV-PKE with the delegation space  $\mathcal{D}$ ).* The main motivation for defining FV-PKE with the delegation  $d$  is to provide the flexibility of verification, which can be used to make the verification result closer to the validity of ciphertexts, as explained below. Let us go back to the motivating example described in the introduction, where a manager asks an assistant to filter out invalid ciphertexts. By using FV-PKE, the manager can give a delegated key  $sk_d$  to the assistant, and the property of verification soundness guarantees that verification using  $sk_d$  can correctly decide the validity for ciphertexts generated by the outsider (i.e., anyone other than the manager and the assistant). However, since the assistant has  $sk_d$ , it does not exclude the possibility that the assistant itself produces ill-formed ciphertexts which are invalid but pass the verification, or are valid but do not pass the verification. We refer to this as an “insider” attack.

Thanks to the fact that FV-PKE supports delegation  $d$ , such “insider” attacks can be easily prevented: the manager can ask several assistants, give them different delegated keys  $(sk_{d_1}, sk_{d_2}, \dots)$ , and regard a ciphertext valid only if it passes all the verifications. As long as not all the assistants collude, it is hard for them to produce ill-formed ciphertexts which are invalid but pass all the verifications, or are valid but do not pass all the verifications. Of course, the manager can also set a threshold, and regard a ciphertext valid if the number of verifications that it passes is above the threshold, in order to tolerate inadvertent errors. This reflects the flexibility of verification. Stepping back, even if an “insider” attack occurs, the manager can identify which assistant produced the ill-formed ciphertexts, by tracing the delegation  $d$  from  $sk_d$ .

Then we formalize the mCCA security for FV-PKE. Compared to the CCA security for standard PKE, we also allow the adversary to obtain delegated keys  $sk_d$  with  $d$  of its choices.

**Definition 10 (mCCA Security of FV-PKE).** *An FV-PKE scheme FPKE is indistinguishable under chosen ciphertext attacks in the multi-challenge setting (mCCA), if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{\text{FPKE}, \mathcal{A}}^{\text{mcca}}(\lambda) := |\Pr[\text{Exp}_{\text{FPKE}, \mathcal{A}, 0}^{\text{mcca}}(\lambda) \Rightarrow 1] - \Pr[\text{Exp}_{\text{FPKE}, \mathcal{A}, 1}^{\text{mcca}}(\lambda) \Rightarrow 1]| \leq \text{negl}(\lambda),$$

where the experiments  $\text{Exp}_{\text{FPKE}, \mathcal{A}, \beta}^{\text{mcca}}(\lambda)$  ( $\beta \in \{0, 1\}$ ) are defined in Fig. 8.

$\text{Exp}_{\text{FPKE}, \mathcal{A}, \beta}^{\text{mcca}}(\lambda): // \beta \in \{0, 1\}$ $\text{pp} \leftarrow \text{Par}(1^\lambda), (pk, sk) \leftarrow \text{Gen}(\text{pp})$ $\mathcal{Q}_{\text{enc}} := \emptyset; \mathcal{Q}_{sk} := \emptyset$ $\beta' \leftarrow \mathcal{A}^{\text{ENC}(\cdot, \cdot), \text{DEC}(\cdot), \text{DELEGATE}(\cdot)}(\text{pp}, pk)$ <p>Output <math>\beta'</math></p> <hr style="border: 0; border-top: 1px solid black; margin: 5px 0;"/> $\text{DELEGATE}(d):$ $sk_d \leftarrow \text{Delegate}(sk, d)$ $\mathcal{Q}_{sk} := \mathcal{Q}_{sk} \cup \{(d, sk_d)\}$ <p>Return <math>sk_d</math></p>	$\text{ENC}(M^0, M^1):$ $ct \leftarrow \text{Enc}(pk, M^\beta)$ $\mathcal{Q}_{\text{enc}} := \mathcal{Q}_{\text{enc}} \cup \{ct\}$ <p>Return <math>ct</math></p> <hr style="border: 0; border-top: 1px solid black; margin: 5px 0;"/> $\text{DEC}(ct):$ <p>If <math>ct \in \mathcal{Q}_{\text{enc}}</math>: return <math>\perp</math></p> <p>Return <math>\text{Dec}(sk, ct)</math></p>
--	---

**Fig. 8.** The IND-mCCA security experiments  $\text{Exp}_{\text{FPKE}, \mathcal{A}, \beta}^{\text{mcca}}(\lambda)$  for FV-PKE.

Finally, we define *ciphertext pseudorandomness* for FV-PKE, which requires the pseudorandomness of ciphertexts for PPT adversaries that are not given any secret key but allowed to access the decryption oracle. This clearly implies anonymity.

**Definition 11 (Ciphertext Pseudorandomness of FV-PKE).** *An FV-PKE scheme FPKE has ciphertext pseudorandomness in the multi-challenge setting, if for any PPT adversary  $\mathcal{A}$ , it holds that*

$$\text{Adv}_{\text{FPKE}, \mathcal{A}}^{\text{cp}}(\lambda) := |\Pr[\text{Exp}_{\text{FPKE}, \mathcal{A}, 0}^{\text{cp}}(\lambda) \Rightarrow 1] - \Pr[\text{Exp}_{\text{FPKE}, \mathcal{A}, 1}^{\text{cp}}(\lambda) \Rightarrow 1]| \leq \text{negl}(\lambda),$$

$\text{Exp}_{\text{FPKE}, \mathcal{A}, \beta}^{\text{cp}}(\lambda): // \beta \in \{0, 1\}$ $\text{pp} \leftarrow \text{Par}(1^\lambda), (pk, sk) \leftarrow \text{Gen}(\text{pp}), \mathcal{Q}_{\text{enc}} := \emptyset$ $\beta' \leftarrow \mathcal{A}^{\text{Enc}(\cdot), \text{Dec}(\cdot)}(\text{pp}, pk)$ <p>Output <math>\beta'</math></p> <hr style="border: 0.5px solid black;"/> $\text{DEC}(ct):$ <p>If <math>ct \in \mathcal{Q}_{\text{enc}}</math>: return <math>\perp</math></p> <p>Return <math>\text{Dec}(sk, ct)</math></p>	$\text{ENC}(M):$ <p>If <math>\beta = 0</math>: <math>ct \leftarrow \text{Enc}(pk, M)</math></p> <p>If <math>\beta = 1</math>: <math>ct \xleftarrow{\\$} \mathcal{CT}</math></p> $\mathcal{Q}_{\text{enc}} := \mathcal{Q}_{\text{enc}} \cup \{ct\}$ <p>Return <math>ct</math></p>
--	--

**Fig. 9.** The ciphertext pseudorandomness experiments  $\text{Exp}_{\text{FPKE}, \mathcal{A}, \beta}^{\text{cp}}(\lambda)$  for FV-PKE, where  $\mathcal{CT}$  denotes the ciphertext space.

where the experiments  $\text{Exp}_{\text{FPKE}, \mathcal{A}, \beta}^{\text{cp}}(\lambda)$  ( $\beta \in \{0, 1\}$ ) are defined in Fig. 9.

**Construction of FV-PKE.** Now we describe our FV-PKE construction as follows. Let  $\Pi = (\Pi.\text{Par}, \Pi.\text{Gen}, \Pi.\text{Prove}, \Pi.\text{MVer}, \Pi.\text{Sim}, \Pi.\text{Delegate}, \Pi.\text{FVer})$  be a tag-based FV-NIZK for linear subspace language  $\mathcal{L}_{[\mathbf{A}]}$  with tag space  $\mathcal{T}$  and delegation space  $\mathcal{D}$ . Let  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathcal{T}$  be a family of collision resistant hash functions. Our FV-PKE construction  $\text{FPKE}_{\text{mcca}} = (\text{Par}, \text{Gen}, \text{Enc}, \text{Dec}, \text{Delegate}, \text{Ver})$  is described in Fig. 10, where the message space is  $\mathbb{G}$  and the delegation space is  $\mathcal{D}$ .

The decryption correctness follows from the completeness (1) of  $\Pi$  and the fact that

$$[v] - \mathbf{w}^\top [\mathbf{c}] = ([\mathbf{w}^\top \mathbf{A}] \mathbf{s} + M) - \mathbf{w}^\top [\mathbf{A} \mathbf{s}] = M,$$

and the verification correctness follows from the completeness (2) of  $\Pi$ .

**Theorem 7 (Verification Soundness of  $\text{FPKE}_{\text{mcca}}$ ).** *If  $\Pi$  is a tag-based FV-NIZK with  $(\delta, \epsilon)$ -verification equivalence, then  $\text{FPKE}$  shown in Fig. 10 has  $(\delta, \epsilon)$ -verification soundness.*

*Proof.* The proof is straightforward. Since  $\Pi$  has  $(\delta, \epsilon)$ -verification equivalence, the algorithms  $\Pi.\text{MVer}$  and  $\Pi.\text{FVer}$  perform identically, except with probability at most  $\epsilon$ . Consequently, it is hard for an (even unbounded) adversary to find  $(ct^*, d^*)$  that passes the verification algorithm  $\text{Ver}$  of  $\text{FPKE}$  (i.e., passing  $\Pi.\text{FVer}$ ) but fails the decryption of  $ct^*$  (i.e., not passing  $\Pi.\text{MVer}$ ), or fails to pass  $\text{Ver}$  (i.e., not passing  $\Pi.\text{FVer}$ ) but decrypts successfully (i.e., passing  $\Pi.\text{MVer}$ ).  $\square$

Now we show that  $\text{FPKE}_{\text{mcca}}$  has almost tight mCCA security and almost tight ciphertext pseudorandomness via the following two theorems.

**Theorem 8 (Almost Tight mCCA Security of  $\text{FPKE}_{\text{mcca}}$ ).** *If the  $\mathcal{D}_{2k, k}$ -MDDH assumption holds in  $\mathbb{G}$ ,  $\mathcal{H}$  is a family of collision resistant hash functions, and  $\Pi$  is a tag-based FV-NIZK with strong USS, then  $\text{FPKE}_{\text{mcca}}$  shown in Fig. 10 is mCCA-secure. Concretely, for any PPT adversary  $\mathcal{A}$ , there exist PPT algorithms  $\mathcal{B}_1, \mathcal{B}_3, \mathcal{B}_3$  s.t.  $\max(\text{Time}(\mathcal{B}_1), \text{Time}(\mathcal{B}_2), \text{Time}(\mathcal{B}_3)) \approx \text{Time}(\mathcal{A}) + (Q_{\text{enc}} + Q_{sk} + Q_{\text{dec}}) \cdot \text{poly}(\lambda)$  with  $\text{poly}(\lambda)$  independent of  $\mathcal{A}$ , and*

$$\text{Adv}_{\text{FPKE}_{\text{mcca}}, \mathcal{A}}^{\text{mcca}}(\lambda) \leq 2\text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + (2k + 4)\text{Adv}_{\mathcal{D}_{2k, k}, \mathbb{G}, \mathcal{B}_2}^{\text{mddh}}(\lambda) + 2\text{Adv}_{\Pi, \mathcal{B}_3}^{\text{uss}}(\lambda) + \frac{6}{q-1},$$

<p><b>Par</b>(<math>1^\lambda</math>):</p> <p><math>\mathbf{A} \leftarrow \mathcal{D}_{2k,k}; H \xleftarrow{\\$} \mathcal{H}</math></p> <p><math>\widehat{\text{pp}} \leftarrow \Pi.\text{Par}(1^\lambda, [\mathbf{A}])</math></p> <p>Return <math>\text{pp} := ([\mathbf{A}], \widehat{\text{pp}}, H)</math></p> <p><b>Gen</b>(<math>\text{pp}</math>):</p> <p><math>\mathbf{w} \leftarrow \mathbb{Z}_q^{2k}</math></p> <p><math>(\text{crs}, \text{td}, \widehat{\text{msk}}) \leftarrow \Pi.\text{Gen}(\widehat{\text{pp}})</math></p> <p>Return <math>pk := ([\mathbf{w}^\top \mathbf{A}], \text{crs}), sk := (\mathbf{w}, \widehat{\text{msk}})</math></p> <p><b>Enc</b>(<math>pk, M \in \mathbb{G}</math>):</p> <p><math>\mathbf{s} \xleftarrow{\\$} \mathbb{Z}_q^k; [\mathbf{c}] := [\mathbf{A}]\mathbf{s} \in \mathbb{G}^{2k}</math></p> <p><math>[v] := [\mathbf{w}^\top \mathbf{A}]\mathbf{s} + M \in \mathbb{G}</math></p> <p><math>\tau := H(pk, [\mathbf{c}], [v])</math></p> <p><math>\pi \leftarrow \Pi.\text{Prove}(\text{crs}, [\mathbf{c}], \mathbf{s}, \tau)</math></p> <p>Return <math>ct := ([\mathbf{c}], [v], \pi)</math></p>	<p><b>Dec</b>(<math>sk, ct = ([\mathbf{c}], [v], \pi)</math>):</p> <p><math>\tau := H(pk, [\mathbf{c}], [v])</math></p> <p>If <math>\Pi.\text{MVer}(\widehat{\text{msk}}, [\mathbf{c}], \tau, \pi) = 1</math> :</p> <p>    Return <math>M' := [v] - \mathbf{w}^\top [\mathbf{c}]</math></p> <p>Otherwise: return <math>\perp</math></p> <p><b>Delegate</b>(<math>sk, d</math>):</p> <p><math>sk_d \leftarrow \Pi.\text{Delegate}(\widehat{\text{msk}}, d)</math></p> <p>Return <math>sk_d</math></p> <p><b>Ver</b>(<math>sk_d, ct = ([\mathbf{c}], [v], \pi)</math>):</p> <p><math>\tau := H(pk, [\mathbf{c}], [v])</math></p> <p>Return <math>\Pi.\text{FVer}(sk_d, [\mathbf{c}], \tau, \pi)</math></p>
--	--

**Fig. 10.** Construction of  $\text{FPKE}_{\text{mcca}}$  from tag-based FV-NIZK  $\Pi$ . For the ease of reading, we emphasize the parts related to  $\Pi$  in gray boxes .

where  $Q_{\text{enc}}, Q_{sk}$  and  $Q_{\text{dec}}$  denote the total numbers of encryption, delegation and decryption queries, respectively.

*Proof.* We prove the theorem via a series of games  $G_0^\beta, \dots, G_5^\beta$  ( $\beta \in \{0, 1\}$ ), where the first two games  $G_0^\beta$  are the mCCA experiments  $\text{Exp}_{\text{FPKE}_{\mathcal{A},\beta}}^{\text{mcca}}(\lambda)$  (cf. Fig. 8), and  $G_5^0, G_5^1$  are identical.

**Game  $G_0^\beta$ .** They are just the original experiments  $\text{Exp}_{\text{FPKE}_{\mathcal{A},\beta}}^{\text{mcca}}(\lambda)$ , except that we use secret key  $\mathbf{w}$  to do the encryption. Due to the equation  $[\mathbf{w}^\top \mathbf{A}]\mathbf{s} = \mathbf{w}^\top [\mathbf{A}\mathbf{s}] = \mathbf{w}^\top [\mathbf{c}]$ , we have that

$$\Pr[\text{Exp}_{\text{FPKE}_{\mathcal{A},\beta}}^{\text{mcca}}(\lambda) \Rightarrow 1] = \Pr[G_0^\beta \Rightarrow 1], \text{ for } \beta \in \{0, 1\}.$$

**Game  $G_1^\beta$ .** In this two games, whenever there is an encryption or decryption query with tag  $\tau'$  that collides with some  $\tau$  used in encryption before, the experiment returns  $\perp$  and aborts. By the collision resistance of  $\mathcal{H}$ , we have

$$|\Pr[G_0^\beta \Rightarrow 1] - \Pr[G_1^\beta \Rightarrow 1]| \leq \text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda), \text{ for } \beta \in \{0, 1\}.$$

**Game  $G_2^\beta$ .** In this two games,  $\text{ENC}(M^0, M^1)$  generates proofs  $\pi$  via  $\Pi.\text{Sim}(\text{td}, \cdot, \cdot)$ .  $G_1^\beta$  and  $G_2^\beta$  are the same due to the perfect zero-knowledge of  $\Pi$ , and we have

$$\Pr[G_1^\beta \Rightarrow 1] = \Pr[G_2^\beta \Rightarrow 1], \text{ for } \beta \in \{0, 1\}.$$

**Game  $G_3^\beta$ .** In this two games, we sample  $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{2k \times k}$  in the beginning of the experiment. Meanwhile,  $\text{ENC}(M^0, M^1)$  computes  $[\mathbf{c}] := [\mathbf{A}_0]\mathbf{s}$ , instead of  $[\mathbf{c}] := [\mathbf{A}]\mathbf{s}$  for  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$ . By the  $\mathcal{D}_{2k,k}$ -MDDH assumption and Lemma 1, we have

$$|\Pr[G_2^\beta \Rightarrow 1] - \Pr[G_3^\beta \Rightarrow 1]| \leq (k+1)\text{Adv}_{\mathcal{D}_{2k,k}, \mathbb{G}, \mathcal{B}_3}^{mddh} + \frac{2}{q-1}, \text{ for } \beta \in \{0, 1\}.$$

**Game  $G_4^\beta$ .** In this two games, the decryption oracle  $\text{DEC}([\mathbf{c}^*], [v^*], \pi^*)$  returns  $\perp$  directly if  $([\mathbf{c}^*], [v^*], \pi^*) \notin \mathcal{Q}_{\text{enc}}$  and  $[\mathbf{c}^*] \notin \mathcal{L}_{[\mathbf{A}]}$ .

Define by **bad** the event that there exists a query  $\text{DEC}([\mathbf{c}^*], [v^*], \pi^*)$ , such that  $([\mathbf{c}^*], [v^*], \pi^*) \notin \mathcal{Q}_{\text{enc}}$ ,  $[\mathbf{c}^*] \notin \mathcal{L}_{[\mathbf{A}]}$ , and there is no hash collision, but  $\Pi.M\text{Ver}(\widehat{msk}, [\mathbf{c}^*], \tau^*, \pi^*) = 1$ , where  $\tau^* := H(pk, [\mathbf{c}^*], [v^*])$ . Obviously,  $G_3^\beta$  and  $G_4^\beta$  are identical unless **bad** happens. Thanks to the strong USS of  $\Pi$ , we have the following lemma.

**Lemma 3** For  $\beta \in \{0, 1\}$ ,  $|\Pr[G_3^\beta \Rightarrow 1] - \Pr[G_4^\beta \Rightarrow 1]| \leq \Pr[\text{bad}] \leq \text{Adv}_{\Pi, \mathcal{B}_4}^{uss}(\lambda)$ .

**Game  $G_5^\beta$ .** In this two games,  $\text{ENC}(M^0, M^1)$  uniformly samples  $[\mathbf{c}] \xleftarrow{\$} \mathbb{G}^{2k}$  and  $[v] \xleftarrow{\$} \mathbb{G}$ , instead of computing  $[\mathbf{c}] := [\mathbf{A}_0]\mathbf{s}$  for  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$  and  $[v] := \mathbf{w}^\top [\mathbf{c}] + M^\beta$ .

**Lemma 4** For  $\beta \in \{0, 1\}$ ,  $|\Pr[G_4^\beta \Rightarrow 1] - \Pr[G_5^\beta \Rightarrow 1]| \leq \text{Adv}_{\mathcal{U}_k, \mathbb{G}, \mathcal{B}_5}^{mddh} + \frac{1}{q-1}$ .

*Proof.* First we argue that in  $G_4^\beta$ ,  $\mathbf{w}$  still contains some entropy which is not leaked via  $pk$  and  $\text{DEC}(\cdot, \cdot, \cdot)$ . Then we show that the left entropy helps us change  $[\mathbf{c}]$  from  $[\mathbf{c}] := [\mathbf{A}_0]\mathbf{s}$  to  $[\mathbf{c}] \xleftarrow{\$} \mathbb{G}^{2k}$ , and change  $[v]$  from  $[v] := \mathbf{w}^\top [\mathbf{c}] + M^\beta$  to  $[v] \xleftarrow{\$} \mathbb{G}$ , based on the  $Q_{\text{sim}}$ -fold  $\mathcal{U}_{2k+1,k}$ -MDDH assumption.

To see this, we redefine  $\mathbf{w}^\top$  as  $\mathbf{w}^\top := \mathbf{w}'^\top + \mathbf{z}^\top \mathbf{A}^\perp$ , where  $\mathbf{w}' \xleftarrow{\$} \mathbb{Z}_q^{2k}$ ,  $\mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^k$ , and  $\mathbf{A}^\perp \xleftarrow{\$} \mathbb{Z}_q^{k \times 2k}$  s.t.  $\mathbf{A}^\perp \mathbf{A} = \mathbf{0}$ . We argue that the information of  $\mathbf{z}$  is totally hidden to  $\mathcal{A}$ .

–  $pk$  hides the information of  $\mathbf{z}$ , due to

$$\mathbf{w}^\top \mathbf{A} = (\mathbf{w}'^\top + \mathbf{z}^\top \mathbf{A}^\perp) \mathbf{A} = \mathbf{w}'^\top \mathbf{A}.$$

–  $\text{DELEGATE}(\cdot)$  hides the information of  $\mathbf{z}$ , since it does not involve  $\mathbf{w}$  at all.  
 –  $\text{DEC}([\mathbf{c}^*], [v^*], \pi^*)$  hides the information of  $\mathbf{z}$ . Thanks to the new rejection rule added in  $G_4$ , we have  $[\mathbf{c}^*] \in \mathcal{L}_{[\mathbf{A}]}$  as otherwise  $\text{DEC}([\mathbf{c}^*], [v^*], \pi^*)$  returns  $\perp$  immediately. Therefore,  $\mathbf{A}^\perp [\mathbf{c}^*] = [\mathbf{0}]$ , and

$$\mathbf{w}^\top [\mathbf{c}^*] = (\mathbf{w}'^\top + \mathbf{z}^\top \mathbf{A}^\perp) [\mathbf{c}^*] = \mathbf{w}'^\top [\mathbf{c}^*].$$

With overwhelming probability we have  $\mathbf{A}^\perp \mathbf{A}_0 \neq \mathbf{0}$ . That is,  $\mathbf{z}^\top \mathbf{A}^\perp \mathbf{A}_0$  is a random value over  $\mathbb{Z}_q^{1 \times k}$  from  $\mathcal{A}$ 's view. According to the  $Q_{\text{sim}}$ -fold  $\mathcal{U}_{2k+1,k}$ -MDDH assumption (equivalently the  $\mathcal{U}_k$ -MDDH assumption due to Lemma 1



and Lemma 2), we know the following two distributions are computationally indistinguishable:

$$\{[\mathbf{A}_0 \mathbf{s}_j], [\mathbf{z}^\top \mathbf{A}^\perp \mathbf{A}_0 \mathbf{s}_j]\}_{j \in [Q_{sim}]} \stackrel{c}{\approx} \{[\mathbf{c}'_j], [v'_j]\}_{j \in [Q_{sim}]},$$

where  $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_q^k$ ,  $\mathbf{c}'_j \xleftarrow{\$} \mathbb{Z}_q^{2k}$ ,  $v'_j \xleftarrow{\$} \mathbb{Z}_q$  for  $1 \leq j \leq Q_{sim}$ .

Recall that in  $\mathbf{G}_4^\beta$ ,  $\text{ENC}(M_0, M_1)$  computes  $[\mathbf{c}], [v]$  as  $[\mathbf{c}] := [\mathbf{A}_0] \mathbf{s}$  and  $[v] := \mathbf{w}^\top [\mathbf{c}] + M^\beta = \mathbf{w}^\top [\mathbf{c}] + M^\beta + \mathbf{z}^\top \mathbf{A}^\perp [\mathbf{A}_0 \mathbf{s}]$ , which are indistinguishable from  $[\mathbf{c}] \xleftarrow{\$} \mathbb{G}^{2k}$  and  $[v] \xleftarrow{\$} \mathbb{G}$  according to the formula above. Then by Lemma 1, Lemma 4 holds as a result.  $\square$

Obviously  $\mathbf{G}_5^0$  and  $\mathbf{G}_5^1$  are identical. At last, thanks to Lemma 2, Theorem 8 follows by taking all things together.  $\square$

**Theorem 9 (Almost Tight Ciphertext Pseudorandomness of  $\text{FPKE}_{\text{mcca}}$ ).** *If the  $\mathcal{D}_{2k,k}$ -MDDH assumption holds in  $\mathbb{G}$ ,  $\mathcal{H}$  is a family of collision resistant hash functions, and  $\Pi$  is a tag-based FV-NIZK with strong USS and proof pseudorandomness, then  $\text{FPKE}_{\text{mcca}}$  shown in Fig. 10 has ciphertext pseudorandomness. Concretely, for any PPT adversary  $\mathcal{A}$ , there exist PPT algorithms  $\mathcal{B}_1, \dots, \mathcal{B}_4$  s.t.  $\max(\text{Time}(\mathcal{B}_1), \dots, \text{Time}(\mathcal{B}_4)) \approx \text{Time}(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \text{poly}(\lambda)$  with  $\text{poly}(\lambda)$  independent of  $\mathcal{A}$ , and*

$$\begin{aligned} \text{Adv}_{\text{FPKE}_{\text{mcca}}, \mathcal{A}}^{\text{cp}}(\lambda) &\leq 2\text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + (2k + 2)\text{Adv}_{\mathcal{D}_{2k,k}, \mathbb{G}, \mathcal{B}_2}^{\text{mddh}}(\lambda) + 2\text{Adv}_{\Pi, \mathcal{B}_3}^{\text{uss}}(\lambda) \\ &\quad + \text{Adv}_{\Pi, \mathcal{B}_4}^{\text{pp}}(\lambda) + \frac{4}{q-1}, \end{aligned}$$

where  $Q_{\text{enc}}$  and  $Q_{\text{dec}}$  denote the total numbers of encryption and decryption queries, respectively.

*Proof.* Theorem 9 is proved via a series of games  $\mathbf{G}_0, \dots, \mathbf{G}_8$ , where  $\mathbf{G}_0$  is the ciphertext pseudorandomness experiment  $\text{Exp}_{\text{FPKE}, \mathcal{A}, 0}^{\text{cp}}(\lambda)$  (cf. Fig. 9), and  $\mathbf{G}_8$  is indistinguishable with  $\text{Exp}_{\text{FPKE}, \mathcal{A}, 1}^{\text{cp}}(\lambda)$ .

Due to the page limitation, we safely omit the descriptions of games  $\mathbf{G}_0, \dots, \mathbf{G}_5$ , since they are similar with those in the proof of Theorem 8.

**Game  $\mathbf{G}_6$ .** In this game, we eliminate the additional check  $[\mathbf{c}^*] \in \text{Span}([\mathbf{A}])$ . Similar to the change from  $\mathbf{G}_3$  to  $\mathbf{G}_4$ , due to the strong USS of  $\Pi$ , we have that

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \text{Adv}_{\Pi, \mathcal{B}_6}^{\text{uss}}(\lambda).$$

**Game  $\mathbf{G}_7$ .** In this game,  $\text{ENC}(M)$  computes  $[\mathbf{c}] := [\mathbf{A}] \mathbf{s}$  for  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$ , instead of  $[\mathbf{c}] \xleftarrow{\$} \mathbb{G}^{2k}$ . By the  $\mathcal{D}_{2k,k}$ -MDDH assumption and Lemma 1, we have

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq k\text{Adv}_{\mathcal{D}_{2k,k}, \mathbb{G}, \mathcal{B}_7}^{\text{mddh}} + \frac{1}{q-1}.$$

**Game  $\mathbf{G}_8$ .** In this game,  $\text{ENC}(M)$  uniformly samples  $[\mathbf{c}] \xleftarrow{\$} \mathbb{G}^{2k}$  and  $\pi \xleftarrow{\$} \mathcal{P}$  instead of  $[\mathbf{c}] := [\mathbf{A}] \mathbf{s}$  for  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$  and  $\pi \leftarrow \Pi.\text{Sim}(\text{td}, [\mathbf{c}], \tau)$ , where  $\mathcal{P}$  denotes the proof space of  $\Pi$ .

**Lemma 5**  $|\Pr[G_7 \Rightarrow 1] - \Pr[G_8 \Rightarrow 1]| \leq \text{Adv}_{\Pi, \mathcal{B}_8}^{pp}(\lambda)$ .

*Proof.* We construct a reduction algorithm  $\mathcal{B}_8$  to distinguish  $\text{Exp}_{\Pi, \mathcal{B}_8, 0}^{pp}(\lambda)$  from  $\text{Exp}_{\Pi, \mathcal{B}_8, 1}^{pp}(\lambda)$  for the proof pseudorandomness security of  $\Pi$  (cf. Fig. 3), as shown in Fig. 11. Recall that  $\mathcal{B}_8$  has access to three oracles SAM, SIM, and VER in  $\text{Exp}_{\Pi, \mathcal{B}_8, \beta}^{pp}(\lambda)$ .

<pre> // <math>\mathcal{B}_8</math> has access to <math>\text{Exp}_{\Pi, \mathcal{B}_8, \beta}^{pp}(\lambda)</math> // for <math>\beta \in \{0, 1\}</math>  <math>\mathcal{B}_8(1^\lambda)</math>: <math>\mathbf{A} \leftarrow \mathcal{D}_{2k, k}</math>; <math>(\widehat{\text{pp}}, \text{crs}) \leftarrow \text{Exp}_{\Pi, \mathcal{B}_8, \beta}^{pp}(\lambda)</math> <math>H \leftarrow \mathcal{H}</math>; <math>\text{pp} := ([\mathbf{A}], \widehat{\text{pp}}, H)</math> <math>\mathbf{w} \leftarrow \mathbb{Z}_q^{2k}</math>; <math>pk := ([\mathbf{w}^\top \mathbf{A}], \text{crs})</math> <math>\mathcal{Q}_{enc} := \emptyset</math>; <math>\mathcal{Q}_\tau := \emptyset</math> <math>\beta' \leftarrow \mathcal{A}^{\text{ENC}(\cdot), \text{DEC}(\cdot)}(\text{pp}, pk)</math> Output <math>\beta'</math>                 </pre>	<pre> ENC(<math>M</math>): <math>[\mathbf{c}] \leftarrow \text{Exp}_{\Pi, \mathcal{B}_8, \beta}^{pp}(\lambda). \text{SAM}(\cdot)</math> <math>[v] \leftarrow \mathbb{G}</math>; <math>\tau := H(pk, [\mathbf{c}], [v])</math> If <math>(\cdot, \cdot, \tau) \in \mathcal{Q}_\tau</math>: return <math>\perp</math> <math>\pi \leftarrow \text{Exp}_{\Pi, \mathcal{B}_8, \beta}^{pp}(\lambda). \text{SIM}([\mathbf{c}], \tau)</math> <math>ct := ([\mathbf{c}], [v], \pi)</math>; <math>\mathcal{Q}_{enc} := \mathcal{Q}_{enc} \cup \{ct\}</math> <math>\mathcal{Q}_\tau := \mathcal{Q}_\tau \cup \{([\mathbf{c}], [v], \tau)\}</math> Return <math>ct</math>  DEC(<math>ct^* = ([\mathbf{c}^*], [v^*], \pi^*)</math>): If <math>ct^* \in \mathcal{Q}_{enc}</math>: return <math>\perp</math> <math>\tau^* := H(pk, [\mathbf{c}^*], [v^*])</math> If <math>\exists([\mathbf{c}], [v], \tau^*) \in \mathcal{Q}_\tau \wedge ([\mathbf{c}], [v]) \neq ([\mathbf{c}^*], [v^*])</math>:     return <math>\perp</math> <math>b \leftarrow \text{Exp}_{\Pi, \mathcal{B}_8, \beta}^{pp}(\lambda). \text{VER}([\mathbf{c}^*], \tau^*, \pi^*)</math> If <math>b = 1</math>: return <math>[v^*] - \mathbf{w}^\top [\mathbf{c}^*]</math> Otherwise: return <math>\perp</math>                 </pre>
--	---

**Fig. 11.**  $\mathcal{B}_8$ 's reduction for the proof of Lemma 5.

Obviously, if  $\mathcal{B}_8$  has access to  $\text{Exp}_{\Pi, \mathcal{B}_8, 0}^{pp}(\lambda)$ , then it simulates  $G_7$  for  $\mathcal{A}$ ; and if  $\mathcal{B}_8$  has access to  $\text{Exp}_{\Pi, \mathcal{B}_8, 1}^{pp}(\lambda)$ , then it simulates  $G_8$  for  $\mathcal{A}$ . Lemma 5 holds as a result.

From  $G_8$  to  $\text{Exp}_{\text{FPKE}, \mathcal{A}, 1}^{pp}(\lambda)$ , we eliminate the additional check of hash collisions in  $\text{ENC}(M)$  and  $\text{DEC}(ct^*)$ . With the same analysis we have

$$|\Pr[G_8 \Rightarrow 1] - \Pr[\text{Exp}_{\text{FPKE}, \mathcal{A}, 1}^{cp}(\lambda) \Rightarrow 1]| \leq \text{Adv}_{\mathcal{H}, \mathcal{B}_8^c}^{cr}(\lambda).$$

Finally, taking Lemma 2 and all things together, Theorem 9 follows.  $\square$

*Remark 11 (Extension to the multi-user setting).* For better readability, we prove the almost tight mCCA security and ciphertext pseudorandomness of  $\text{FPKE}_{\text{mcca}}$  in the single-user setting in Theorems 8 and 9. Now we show how to extend the proof techniques to the multi-user setting. More precisely, the public parameter  $\text{pp} = ([\mathbf{A}], \widehat{\text{pp}}, H)$  is shared among all users, and each user  $i \in [\mu]$  samples its own master secret key  $(\mathbf{w}^{(i)}, \widehat{\text{msk}}^{(i)})$ . In all computational steps in the proof, we

modify all samples of  $[c]$  simultaneously, based on the random self-reducibility of the MDDH assumption. Moreover, the underlying FV-NIZK scheme  $\Pi$  is required to have almost tight strong USS and proof pseudorandomness in the multi-user setting, which is satisfied by the first construction in Subsect. 4.1.

**Acknowledgments.** We would like to thank the anonymous reviewers for their valuable comments and suggestions. Shengli Liu and Xiangyu Liu were partially supported by National Natural Science Foundation of China (NSFC No. 61925207), Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), and the National Key R&D Program of China under Grant 2022YFB2701500. Shuai Han was partially supported by National Natural Science Foundation of China (Grant No. 62002223), Shanghai Sailing Program (20YF1421100), Young Elite Scientists Sponsorship Program by China Association for Science and Technology (YESS20200185), and Ant Group through CCF-Ant Research Fund (CCF-AFSG RF20220224). Dawu Gu is partially supported by the National Key Research and Development Project (Grant No. 2020YFA0712302).

## References

1. Abdalla, M., Bourse, F., Caro, A.D., Pointcheval, D.: Simple functional encryption schemes for inner products. In: PKC 2015, vol. 9020, pp. 733–751 (2015)
2. Abdalla, M., Bourse, F., Caro, A.D., Pointcheval, D.: Better security for functional encryption for inner product evaluations. IACR Cryptol. ePrint Arch. **2016**, 11 (2016)
3. Abe, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Tagged one-time signatures: tight security and optimal tag size. In: PKC 2013, vol. 7778, pp. 312–331 (2013)
4. Abe, M., Jutla, C.S., Ohkubo, M., Pan, J., Roy, A., Wang, Y.: Shorter QA-NIZK and SPS with tighter security. In: ASIACRYPT 2019, vol. 11923, pp. 669–699 (2019)
5. Abe, M., Jutla, C.S., Ohkubo, M., Roy, A.: Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In: ASIACRYPT 2018, pp. 627–656 (2018)
6. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: CRYPTO 2016, pp. 333–362 (2016)
7. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: CRYPTO 1989, vol. 435, pp. 194–211 (1989)
8. Blazy, O., Kakvi, S.A., Kiltz, E., Pan, J.: Tightly-secure signatures from chameleon hash functions. In: PKC 2015, pp. 256–279 (2015)
9. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: CRYPTO 2014, pp. 408–425 (2014)
10. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: TCC 2011, vol. 6597, pp. 253–273 (2011)
11. Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: CRYPTO 2013, vol. 8043, pp. 435–460 (2013)
12. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: EUROCRYPT 2002, vol. 2332, pp. 45–64 (2002)

13. 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)
14. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: *STOC 1991*, pp. 542–552 (1991)
15. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for Diffie-Hellman assumptions. In: *CRYPTO 2013*, vol. 8043, pp. 129–147 (2013)
16. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: *EUROCRYPT 2016*, vol. 9665, pp. 1–27 (2016)
17. Gay, R., Hofheinz, D., Kohl, L.: Kurosawa-desmedt meets tight security. In: *CRYPTO 2017*, vol. 10403, pp. 133–160 (2017)
18. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. *J. ACM* **59**(3), 1–35 (2012)
19. Han, S., et al.: Authenticated key exchange and signatures with tight security in the standard model. In: *CRYPTO 2021*, vol. 12828, pp. 670–700 (2021)
20. Han, S., Liu, S., Lyu, L., Gu, D.: Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In: *CRYPTO 2019*, vol. 11693, pp. 417–447 (2019)
21. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. *Des. Codes Cryptogr.* **80**(1), 29–61 (2016)
22. Hofheinz, D., Jia, D., Pan, J.: Identity-based encryption tightly secure under chosen-ciphertext attacks. In: *ASIACRYPT 2018*, vol. 11273, pp. 190–220 (2018)
23. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: *ASIACRYPT 2013*, vol. 8269, pp. 1–20 (2013)
24. Libert, B., Joye, M., Yung, M., Peters, T.: Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In: *ASIACRYPT 2014*, pp. 1–21 (2014)
25. Libert, B., Peters, T., Joye, M., Yung, M.: Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In: *ASIACRYPT 2015*, vol. 9452, pp. 681–707 (2015)
26. Liu, X., Liu, S., Han, S., Gu, D.: Tightly CCA-secure inner product functional encryption scheme. *Theor. Comput. Sci.* **898**, 1–19 (2022)
27. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: *STOC 1990*, pp. 427–437 (1990)
28. O’Neill, A.: Definitional issues in functional encryption. *IACR Cryptol. ePrint Arch.* **2010**, 556 (2010)
29. Ràfols, C.: Stretching Groth-Sahai: NIZK proofs of partial satisfiability. In: *TCC 2015*, vol. 9015, pp. 247–276 (2015)
30. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: *FOCS 1999*, pp. 543–553 (1999)
31. Tomida, J.: Tightly secure inner product functional encryption: multi-input and function-hiding constructions. In: *ASIACRYPT 2019*, pp. 459–488 (2019)



# Zero-Knowledge Arguments for Subverted RSA Groups

Dimitris Kolonelos<sup>1,2(✉)</sup>, Mary Maller<sup>3</sup>, and Mikhail Volkhov<sup>4</sup>

<sup>1</sup> IMDEA Software Institute, Madrid, Spain  
`dimitris.kolonelos@imdea.org`

<sup>2</sup> Universidad Politecnica de Madrid, Madrid, Spain

<sup>3</sup> Ethereum Foundation, London, UK  
`mary.maller@ethereum.org`

<sup>4</sup> The University of Edinburgh, Edinburgh, UK  
`mikhail.volkhov@ed.ac.uk`

**Abstract.** This work investigates zero-knowledge protocols in subverted RSA groups where the prover can choose the modulus and where the verifier does not know the group order. We introduce a novel technique for extracting the witness from a general homomorphism over a group of unknown order that does not require parallel repetitions. We then present a NIZK range proof for general homomorphisms as Paillier encryptions in the designated verifier model that works under a subverted setup. The key ingredient of our proof is a constant sized NIZK proof of knowledge for a plaintext. Security is proven in the ROM assuming an IND-CPA additively homomorphic encryption scheme. The verifier’s public key can be maliciously generated and is reusable and linear in the number of proofs to be verified.

## 1 Introduction

A zero-knowledge proof consists of a prover that demonstrates to a verifier that a statement is true while revealing no information about the witness. Sigma protocols [28, 58] are a special type of zero knowledge proof that avoid expensive NP encodings and work naturally with many popular non-general relations. Sigma protocols enjoy negligible soundness-error in groups of known order. The story is different in groups of hidden order where negligible soundness can only be achieved by running  $O(\lambda)$  sigma protocols in parallel [6, 60], thus multiplying the prover, proof size, and verifier costs by  $O(\lambda)$ .

In the common reference string model [11], a negligible soundness-error of hidden order group sigma protocols can be directly linked to hardness assumptions such as the strong-RSA [9, 27, 34, 40]. However, relying on hardness assumptions introduces an avenue for subversion: we can make no guarantees about any hardness assumption when a malicious prover corrupts the parameters of the hidden

---

D. Kolonelos and M. Volkhov—Most of the work was done while the first and third authors were interns at Ethereum Foundation.

order group. For the prominent case of RSA-groups, i.e., multiplicative groups over the ring  $\mathbb{Z}_N$  with  $N = p \times q$ , subversion is easy because one can compute the order of the group given the factorization,  $p$  and  $q$ .

To date, no natural<sup>1</sup> protocol for general homomorphism-languages with hidden order co-domain has negligible soundness-error (without repetitions), and at the same time does not rely on computational assumptions over the co-domain. Indeed, the task of constructing zero-knowledge proofs over subverted RSA-groups is exceedingly challenging; strictly more so than over traditional hidden order groups that are correctly formed. One can make no guarantees about how the modulus was generated and the Fiat-Shamir challenges can be continuously sampled until one from a malicious distribution is found.

**Our Question.** We thus put forward the question:

*Can one build a generalised sigma-protocol in subverted RSA-groups achieving negligible soundness-error without repetitions?*

Our answer to this question is affirmative assuming a designated-verifier; we provide and prove secure a construction in the designated verifier model [33, 55]. This is excellent news because currently the only known method to construct RSA-groups is via a trusted setup [45]. Generating secure RSA parameters with a MPC is an extremely challenging task to realise in practice and to date no large scale RSA-MPCs have ever been completed. Our work thus provides an exciting avenue for numerous results in RSA-groups to remain applicable in subverted settings.

Subverted RSA groups are primarily interesting because they are a rare instantiation for groups of unknown order. The only known alternative for building hidden order groups is class groups, that can also be used to build ZKPs (e.g. [26]). In high contrast to RSA groups, cryptanalysts have only recently started focusing on class groups and we are still learning the best practices for choosing the parameters for implementation [38, 47, 50].

Further, the potential for  $N$  to be subverted is a delicacy which is rarely considered when using the additively homomorphic Paillier [54] encryption scheme. Here subverted parameters should be considered the default because participants can choose their encryption modulus  $N$ . Nonetheless, the handling of subverted parameters is a detail that is often overlooked in protocols that use Paillier. For example, in the influential paper by Hazay et al. [45], we see that they require a subversion resistant zero-knowledge range proof to realise their multiparty MPC but that none of their suggestions are subversion resistant. For more detail see the full version of the paper. As a second example, in the Damgard-Jurik voting scheme [36], they assume that a modulus  $N$  is generated by a trusted third party. If it were instead chosen by an election authority—which is a likelihood in real world systems—then this modulus could certainly be subverted. By colluding

---

<sup>1</sup> By ‘natural’ we mean a protocol that works directly for the underlying language and does not involve NP-reductions.

with just a single voter, the authority could provide verifying proofs of faulty encryptions and thus entirely decide the election result.

## 1.1 Our Contributions

In this paper we investigate zero-knowledge proofs under subverted RSA parameters. This is an extremely adversarial setting where the modulus  $N$  can be factorised by the prover but not by the verifier. We make no assumptions about ideal properties of the modulus: for example we can have that  $N$  is smooth or even that the prover knows the factorisation of  $N$ .

Our first contribution is a *new extraction method* for extracting a witness inside general homomorphisms. This extraction technique is completely new to the literature. We reify this technique through a designated-verifier protocol, named  $DV_{\text{Prot}}$ , which answers affirmatively the main question of this work introduced in the previous section. A substantial caveat for our extractor is that the challenges used by the sigma protocol are encrypted (under the designated verifiers secret key which importantly is independent from the potentially subverted  $N$ ). Our extractor should fail if the adversary could decrypt the challenges, thus we describe the general extraction method and reduce the probability of the extractor failing to an adversary's advantage against IND-CPA. At the heart of our extraction method is an information-theoretical lemma about the distribution of the challenges extracted, which we prove to hold unconditionally. Exemplifying the extraction method, and as a stepping stone towards the second contribution, we explain how to make the  $DV_{\text{Prot}}$  protocol practical, with reusable and potentially maliciously generated verifier's public key. Our main results are in the random oracle model however we also provide an optimised version in the generic group model.

Using our extraction technique we arrive at our second contribution, namely a zero-knowledge designated verifier *range proof* for Paillier encryptions under subverted modulus with negligible soundness, which we call  $DVRange_{\text{Prot}}$ . The protocol prevents a prover from encrypting a value outside the range even if the prover chooses the encryption key. Our proof is non-interactive (in the random oracle model) and has negligible soundness error without parallel repetitions. Security is proven in the RO model under the assumption that Paillier is IND-CPA. Our techniques for proving security are potentially of independent interest and described in more detail in Sect. 1.3. In the full version we show how our range proof can be applied for non-injective homomorphisms.

The verifier's public key has size  $\mathcal{O}((\lambda + Q) \log N)$  for  $N$  a Paillier modulus,  $\lambda$  the security parameter, and  $Q$  the number of proofs the verifier will respond to. Our protocol does not require a common reference string; being DV the (designated) verifier inherently runs a setup to generate their potentially malicious key. To ensure zero-knowledge holds against all verifier keys we describe a non-interactive publicly verifiable key generation algorithm. In more detail, the verifier runs a publicly verifiable range proof to demonstrate that the verification public key (VPK) contains ciphertexts in the correct range. We apply amortisation techniques by Cramer et al. [30] (in Sect. 4.3) to minimise the cost

of this range proof. The key generation process is relatively expensive and can be avoided in scenarios where the verifier only needs to retrospectively prove honest behaviour by revealing the secrets behind their public key. Such scenarios are common in applications such as MPC with identifiable abort (ID-MPC, [46]).

## 1.2 Related Work

In composite order groups the standard  $\Sigma$ -protocol has knowledge error of only  $1/2$  [6]. For a negligibly small extraction error one needs to run the protocol  $\lambda$  times in parallel (for  $\lambda$  the security parameter). This induces an  $O(\lambda)$  multiplicative overhead. There are many different approaches in the literature to proving composite group statements more efficiently which we summarise here.

**Proofs over Groups of Unknown Order.** An intensive line of work focuses on constructing efficient zero knowledge proofs for relations over groups where the order is unknown to all parties, however none would fit our context. The Fujisaki-Okamoto solution [27,34,40], the protocols of [8,18] and the solution by Boneh et al. [13] being computationally-sound are not sound in subverted RSA groups because having known (to the prover) group order prevents the underlying computational assumptions from holding. The protocol of [7] considers a model where the verifier has extra information about the witness<sup>2</sup>. The protocol from [5] was later cryptanalyzed [49]. For specific relations, [35,36] present efficient protocols where the prover knows the order of the group, however they are sound only when the RSA group is correctly formed. The work of Cramer et al. [29,30] presents a transformation that allows the protocol to have negligible soundness error, yet only when proving  $\lambda$  statements simultaneously. For a single proof it cannot be applied. Finally, Bangerter et al. [6] and Terelius et al. [60] show a lower bound on soundness error for constant round sigma-like protocols in the standard model (no CRS, no RO), that translates to  $1/2$  for common parameters.

**Proving RSA Relations with zk-SNARKs.** Many zk-SNARK proof systems are both general enough to encode any NP circuit and efficient enough to be used in practice. Thus we can prove relations about subverted RSA groups by representing them with an arithmetic circuit or similar. Ozdemir et al. implement an RSA based accumulator inside a SNARK [53]. Their work improves upon xJsnark [48]. Using Ozdemir et al.'s BigNat library<sup>3</sup> we compute the size of the Paillier knowledge-of-plaintext circuit at 80 million gates for 2048 bit  $N$ . This is towards the upper end of what can feasibly be computed with a SNARK. To the best of our knowledge the biggest circuits currently in production have about 100-million constraints and take minutes to compute even on specialist hardware<sup>4</sup>. Our work does not require a reduction to NP and therefore we avoid

<sup>2</sup> For some relations (e.g. Paillier Encryptions) this can lead to fully reconstructing the witness.

<sup>3</sup> <https://github.com/alex-ozdemir/bellman-bignat>.

<sup>4</sup> <https://research.protocol.ai/sites/snarks/>.



this prover overhead. Our approach also avoids the significant challenge of auditing an 80 million gate circuit.

**Range Proofs in the RSA Setting.** In this work we present range proofs for RSA-like relations (e.g. Paillier encryption), or generally (additive) homomorphisms with unknown co-domain. Variations of basic Schnorr-like  $\Sigma$ -protocol exist for RSA-like range relations [12, 18, 20, 25, 27, 34, 39]. Boudot [14] presents the first range proof for general range  $[L, R]$  with slackness 1 (i.e. the message lies exactly in  $m \in [0 \dots R]$  as opposed to some extended range  $m \in [0 \dots \delta R]$ ). Further [14] uses a so-called four-squares integer decomposition property, a technique which is later used and improved in [44, 51, 62]. None of these works consider a subverted modulus. In fact they are computationally sound and make assumptions about the RSA group, thus they do not work in subverted settings.

**Proofs of Correct Form of Moduli.** An orthogonal to the above line of work intends to prove that the group itself is not subverted [3, 10, 19, 41, 42, 61], meaning that the modulus  $N$  of the RSA group has some beneficial property; for example is square-free, a product of two primes, a product of equally-sized primes, a Blum integer or a product of two safe primes, etc. Other works consider proving that moduli are correctly formed in the context of specific applications as password-based key agreement [23] or threshold ECDSA signatures [21]. All these solutions require repetitions to reach a negligible soundness-error. Furthermore, to apply computationally-sound protocols for general homomorphisms (such as Fujisaki-Okamoto) over the group afterwards, one needs to prove that the RSA group is a product of two safe primes. Only [19] ensures this, however it has high costs and does not avoid the  $O(\lambda)$  parallel repetitions.

### 1.3 Overview of Techniques

In this work we design efficient designated-verifier ZK protocols for knowledge and range of RSA group homomorphisms, which have negligible soundness error without repetitions even when the group is maliciously chosen. The main unifying ideas of all our techniques are (1) an alternative approach to  $\Sigma$ -protocols' witness extraction and (2) a careful realisation through homomorphic encryption with respect to (also potentially subverted) verifier's modulus, which allows hiding protocol challenges from the prover in a way that prevents lower-bound attacks of [6, 60].

Let  $\psi : \mathcal{D} \rightarrow \mathbb{H}$  be a group homomorphism where  $\mathbb{H}$  is an RSA-related group, such as exponentiations  $w \mapsto g^w$  over  $\mathbb{H} = \mathbb{Z}_N^*$  (or multiexponentiations), or Paillier encryption  $(w, r) \mapsto (N+1)^w h^r$ . We wish to design an efficient argument of knowledge of  $w$  such that  $Y = \psi(w)$ , and  $w \in \{0 \dots R\}$  for  $R \in \mathcal{D} \subset \mathbb{Z}$ .

**$\Sigma$ -Protocol Soundness.** The classic  $\Sigma$ -protocol for proving knowledge of  $w$  such that  $Y = \psi(w)$ , described in Sect. 1, is only secure if elements from  $\mathcal{D}$  are invertible. The standard special-soundness extractor behaves as follows: given

two successful transcripts with the same first message  $(a, c, s), (a, c', s')$  such that  $aY^c = \psi(s)$  and  $aY^{c'} = \psi(s')$  and  $c \neq c'$  it combines the two:

$$aY^c = \psi(s) \quad aY^{c'} = \psi(s')$$

from which it gets  $Y = \psi(s-s')^{(c-c')^{-1}} = \psi((s-s')(c-c')^{-1})$ . When  $\mathbb{H}$  is a group of public prime order  $p$ , as in case of the Schnorr protocol, this strategy always succeeds, because  $(c - c')^{-1} \pmod p$  is efficiently computable. However, when  $\mathbb{H}$  is a maliciously chosen RSA group, the extractor has two problems. First, it does not know the order of the group and thus can only compute  $(c - c')^{-1}$  when  $c - c' = 1$  (in this trivial case  $Y^1 = \psi(s - s')$ , and  $s - s'$  is the witness). This limitation is similar to the hardness of taking roots in groups of unknown order. Second, some inverses  $(c - c')^{-1}$  do not exist because it is possible that  $\gcd(c - c', \text{ord}(\mathcal{D})) \neq 1$  for a maliciously chosen  $N$ .

In fact the impossibility results of [6,60] show that the above extractor fails for any group  $\mathbb{H}$  whose order is not publicly known, such as RSA groups.

**A Generalized Extraction Lemma.** Towards constructing an efficient protocol with negligible soundness error, our starting point is a generalized extraction approach. Assume that our extractor has  $M \geq 3$  successful transcripts<sup>5</sup>  $\{(a, c_i, s_i)\}_{i=1}^M$  such that:

$$aY^{c_1} = \psi(s_1) \quad aY^{c_2} = \psi(s_2) \quad \dots \quad aY^{c_M} = \psi(s_M)$$

then combining the first with the rest we get the equivalent:

$$Y^{c_2-c_1} = \psi(s_2 - s_1) \quad \dots \quad Y^{c_M-c_1} = \psi(s_M - s_1)$$

Now if  $\gcd(c_2 - c_1, \dots, c_M - c_1) = 1$  then we can always compute coefficients  $\gamma_2, \dots, \gamma_M$  such that  $\gamma_2(c_2 - c_1) + \dots + \gamma_M(c_M - c_1) = 1$ , which means:

$$Y^1 = Y^{\gamma_2(c_2-c_1)+\dots+\gamma_M(c_M-c_1)} = \psi(\gamma_2(s_2 - s_1) + \dots + \gamma_M(s_M - s_1))$$

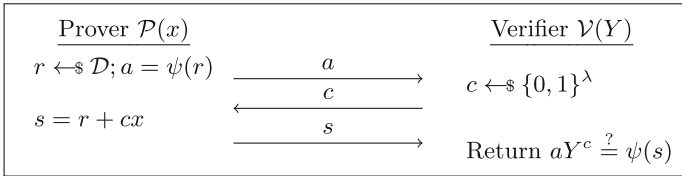
so  $s^* = \gamma_2(s_2 - s_1) + \dots + \gamma_M(s_M - s_1)$  is a valid pre-image.

This extraction technique succeeds as long as  $\gcd(c_2 - c_1, \dots, c_M - c_1) = 1$ . If we had an honest prover and the  $c_i$  challenges were truly random and independent, then well-known results from mathematics show that this happens with probability  $1/\zeta(M)$ , for  $\zeta$  being the zeta Riemann function. This probability is overwhelming (negligibly close to 1) as a function of  $M$ .

However, a malicious prover may choose not to respond upon receiving certain challenges  $c$ , so that  $\gcd(c_2 - c_1, \dots, c_M - c_1) \neq 1$ . As an example they can choose only to answer even challenges. The natural conclusion is that for this generalized extraction to work we need the (adversarial) prover to be oblivious to the challenges it answers.

---

<sup>5</sup> Extracting  $k$  successful transcripts is no harder than extracting 2 [1].



**Fig. 1.** A  $\Sigma$ -protocol for the relation containing elements  $(Y, w)$  such that  $Y = \psi(w)$ , where  $\psi$  is a general homomorphism. This protocol is only knowledge sound if elements from  $\mathcal{D}$  are invertible.

**Designated Verifier Techniques.** We bootstrap the protocol of Fig. 1 to a secure one (with negligible soundness error) in the Designated-Verifier model.

One of our key observations is that in the Designated-Verifier setting we can hide the challenge  $c$  from the malicious prover by encrypting it with a homomorphic encryption scheme for verifier’s public key. Then the prover computes the response to the challenge “blindly”, using additive homomorphism of the encryption scheme. The verifier, who possesses the secret key of the encryption, decrypts the response normally in order to retrieve the plaintext response of the  $\Sigma$ -protocol. For this we need the verifier to hold the corresponding secret key, which must be kept secret from the prover. The public key of the designated verifier (VPK) is merely the  $\text{pk}$  of the encryption scheme and the ciphertext  $\text{ct}$  of the encrypted challenge. The idea of encrypting a (single) challenge in the designated-verifier public key appears in previous DV protocols [24, 33]

To prove the existence of an extractor we require  $M$  answers with different challenges from the prover. This is clearly not possible when we encrypt just a single challenge; but we also cannot do it even when we encrypt  $M$  challenges—the prover can potentially choose only to answer with respect to the first challenge. What we require is an exponential sized challenge space. For this, we encrypt  $\lambda$  sub-challenges that are chosen uniformly at random:  $\text{ct}_1 = \text{Enc}(c_1), \dots, \text{ct}_\lambda = \text{Enc}(c_\lambda)$  and add them to the public key. Then the value  $\mathcal{P}$  responds to is a random  $(0, 1)$  linear combination of  $\{c_i\}$ :  $c = \sum_{i=1}^\lambda b_i c_i$  where  $\mathbf{b} = (b_1, \dots, b_\lambda)$  a random bitstring-challenge sampled by the verifier, which gives rise to exponential  $\mathcal{C}$ .

To prove soundness, the core of our security proof is an information-theoretical lemma showing that after  $M = \text{poly}(\lambda)$  linear combinations have been extracted, the probability of  $\{\mathbf{b}_i \mathbf{c}^\top\}_{i=1}^M$  being coprime is overwhelming (assuming that  $c_i$ ’s were uniformly sampled and independent during the setup).

**DV with a Reusable VPK.** A common issue in the Designated-Verifier model is that a prover, after seeing whether some proofs of its choice verify or not, can learn information about the VPK’s structure and break soundness. This is the analogue of IND-CCA security of encryption schemes. Intuitively, the verification oracle behaves in a similar manner to a decryption oracle. Additive homomorphic encryption schemes cannot be IND-CCA and thus an attacker

could use a verification oracle to learn information about  $\text{vpk}$ . We overcome this by adding  $Q = \text{poly}(\lambda)$  statistical blinding factors  $e_1, \dots, e_Q$  encrypted in the VPK. At each proof one of these factors is added to the linear combination and thus statistically blinds it; thus  $Q$  is maximum number of verification queries the prover can ask. The CRS size is thus  $O(1)$  per proof.

#### 1.4 Comparison with Alternative Approaches

To the best of our knowledge, this work is the first that deals with the problem of constructing zero-knowledge proofs in subverted RSA groups. On the other hand, the literature provides numerous techniques on constructing zero-knowledge proofs in non-subverted RSA groups. It is challenging to compare the efficiency of our scheme directly against the state-of-the-art for non-subverted solutions because this would require fully researching how to convert multiple solutions into the subverted setting. Instead we here briefly justify our techniques against two possible alternative approaches that provide partial solutions to the problem.

**Combine with an Auxiliary Group of Unknown Order.** A possible approach to constructing a sound *proof of knowledge* in the subverted RSA setting would be to combine the simple protocol of Fig. 1 with a proof of a preimage in an established group of unknown order. That is, generate an unknown order group  $\mathbb{G}$ , commit to the same preimage  $\text{Commit}(w)$  and send the commitment to the verifier. Then compose in parallel a proof of knowledge for  $\text{Commit}(w)$  (over  $\mathbb{G}$ ) and the protocol of Fig. 1 (over the subverted RSA group). The Fujisaki-Okamoto extraction technique [27, 34, 40] gives negligible knowledge error and avoids the need for  $\lambda$  repetitions. However, this solution either requires a private-coin trusted setup in case an RSA group is used as the auxiliary group of unknown order, or must rely on class groups [16]. Solutions relying on class groups are outside the scope of this work (see Introduction).

**Range Proof with an Auxiliary Prime Order Group.** For the *range proof* problem for the preimage  $w$  of a homomorphism,  $Y = \psi(w)$  with  $0 < w < R$ , one possible approach is the following. Generate an auxiliary prime order group  $\mathbb{G}$  and commit to the preimage,  $\text{Commit}(w)$  over this group (e.g. via Pedersen commitment). Then run in parallel the protocol of Fig. 1 for  $\psi(w)$  in the subverted RSA group and a simple Schnorr protocol for the commitment on  $\mathbb{G}$ , to prove that  $\text{Commit}(w)$  and  $\psi(w)$  contain the same value. Afterwards one can use a range proof protocol in the prime order group [17, 26] to prove the range of  $w$ . The main benefit here is that due to progress on range proofs over prime order groups, the actual range proof block is concretely efficient.

This solution, however, inherits the soundness-error (and thus the required iterations) of the protocol of Fig. 1. That is  $1/2$  for general homomorphisms  $1/\text{poly}(\lambda)$  for some specific special homomorphisms such as the (original) Paillier Encryption [6]. This leads to an overhead of  $O(\lambda)$  and  $O(\lambda/\log(\lambda))$  respectively, due to the repetitions needed.

Our work concerns with the former category, general non-special homomorphisms (such as ElGamal-Paillier) where the overhead is  $O(\lambda)$ , and provides a truly unique perspective on how to decrease their asymptotic efficiency to  $O(1)$  which was not previously known to be possible. We achieve this by providing and proving secure an alternative extraction technique together with an information theoretical lemma that have no dependence on parallel executions.

## 2 Preliminaries

### 2.1 Notation

We denote the security parameter with  $\lambda$ ;  $\text{poly}(\lambda)$  is any positive  $f(n) = O(\text{poly}(n))$ , and  $\text{negl}(\lambda)$  is a negligible positive function. With  $[a, b]$  we denote the set  $\{a, a + 1, \dots, b\}$ , and with  $[n]$  we denote  $[1, n]$ . Similarly with  $\llbracket n \rrbracket$  we denote the set  $[-\lfloor \frac{n}{2} \rfloor \dots \lfloor \frac{n}{2} \rfloor]$ . Adversaries are assumed to be stateful unless stated otherwise.

$\mathbb{Z}_n$  is the additive group of order  $n$ . We often explicitly consider interval  $\llbracket n \rrbracket$  as the integer encoding for  $\mathbb{Z}_n$ .  $\mathbb{Z}_n^*$  is the multiplicative group of all integers in  $\llbracket n \rrbracket$  coprime with  $n$ . With  $\phi(\cdot)$  we denote the Euler's totient function.  $\mathcal{U}_S$  stands for uniform distribution on  $S$  as a finite set (e.g.  $\mathcal{U}_{\mathbb{Z}_p}$ );  $\mathcal{U}_{[L,R]}$  is a uniform distribution on  $[L, R]$ , and  $\mathcal{U}_R$  is a shorthand for  $\mathcal{U}_{[0,R]}$ . In general we denote with capital letters, e.g.  $Y$ , elements of the RSA group. In bold we denote vectors (e.g.  $\mathbf{s}$ ) and matrices (e.g.  $\mathbf{A}$ ).

### 2.2 Homomorphic Encryption Schemes

In this work we engage public-key encryption schemes that have additively homomorphic properties. That is an encryption scheme is called additively homomorphic if for every  $\text{pk} \in \mathcal{PK}$  and  $m_1, m_2 \in \mathcal{M}$ ,  $\text{Enc}_{\text{pk}}(m_1) \cdot \text{Enc}_{\text{pk}}(m_2) = \text{Enc}_{\text{pk}}(m_1 + m_2)$ , where  $\cdot$  is a ciphertext space operation. In the rest we assume that the message space  $\mathcal{M}$  of the additively homomorphic schemes we refer to forms a ring. Some known examples of additively homomorphic encryption are the Paillier cryptosystem and its variants [15, 31, 36, 54] in the RSA setting, the Castagnos-Laguillaumie cryptosystem over class groups [22] and schemes from lattices [43, 56]. Notably, no additively homomorphic public-key cryptosystems from groups of prime order exist.<sup>6</sup>

**Paillier Encryption Scheme.** We briefly recall the Paillier public key encryption scheme [54], and refer the reader to our full version for more details.

---

<sup>6</sup> Although the lifted ElGamal cryptosystem (alike ElGamal but the message is lifted in the exponent) is additively homomorphic, the decryption is not polynomial-time, unless one restricts the message space to polynomial size. This makes it unsuitable for most applications.

**KeyGen**( $1^\lambda$ ): sample  $p, q$  primes of the size  $\lambda$  and set  $N = p \cdot q$ . Compute  $d = \phi(N)^{-1} \bmod N^2$ . Output  $\text{pk} = N$  and  $\text{sk} = (d, \phi(N))$ .  
**Enc** $_{\text{pk}}$ ( $m$ ): sample uniformly  $r \leftarrow_s \mathbb{Z}_N^*$  and output  $\text{ct} = (N + 1)^m r^N \bmod N^2$ .  
**Dec** $_{\text{sk}}$ ( $\text{ct}$ ): compute  $c = (\text{ct}^{\phi(N)} - 1)d \bmod N^2$  and return  $m = \frac{c}{N}$ .

### 2.3 Homomorphisms and Efficient $\Sigma$ -protocols

Let  $\psi : \mathcal{D} \rightarrow \mathbb{H}$  be a homomorphism between a domain  $\mathcal{D}$  (group or ring), and an output group  $\mathbb{H}$  (e.g. RSA). When  $Y = \psi(w)$ , we call  $w$  a witness, and  $Y$  an instance.

A pair  $(v, u) \in \mathbb{Z} \times \mathcal{D}$  is called a *pseudo-preimage* (PP) for instance  $Y = \psi(x)$ , if  $Y^v = \psi(u)$  holds [5, 7], where  $v$  is called a degree of a given PP. Pseudo-preimages naturally occur in  $\Sigma$ -protocols: the extractor usually transforms two transcripts for the same commitment  $a$  ( $Y^{c_i} a = \psi(s_i)$ ,  $i \in 1, 2$ ) into a single PP by dividing the equations:  $Y^{c_1 - c_2} = \psi(s_1 - s_2)$ , thus  $(c_1 - c_2, s_1 - s_2)$  is a PP.

In prime-order groups ( $|\mathbb{H}| = p$ ) knowledge of PP implies knowledge of preimage, since inverses in  $\mathbb{Z}_p$  are efficiently computable. In groups where the order is not prime or even unknown to  $\mathbb{V}$  (e.g. in Paillier  $\mathbb{H} = \mathbb{Z}_{N^2}^*$ ) there is another way to extract a proper preimage, but from *two* pseudo-preimages: given  $(v_1, u_1)$ ,  $(v_2, u_2)$  with  $\gcd(v_1, v_2) = 1$  for any  $Y$  we can use the so-called called “Shamir’s trick”. Given  $(v_1, u_1), (v_2, u_2)$  s.t.  $Y^{v_i} = \psi(u_i), i \in \{1, 2\}$ , it first checks if  $\gcd(v_1, v_2) \neq 1$  and aborts if not. Then it computes Bezout coefficients—integers  $\gamma, \delta$  such that  $\gamma v_1 + \delta v_2 = 1$ , and returns  $u := \gamma u_1 + \delta u_2$ . This extractor succeeds, since given  $Y^{v_i} = \psi(u_i)$ ,  $Y = Y^{\gamma v_1 + \delta v_2} = \psi(u_1 \gamma + u_2 \delta) = \psi(u)$ .

**Special Homomorphisms.** In [7], following Cramer [28], the homomorphism  $\psi : \mathcal{D} \rightarrow \mathbb{H}$  is called *special* if for any instance  $Y$  one can easily find a *non-trivial* PP  $(\hat{v}, \hat{u})$  of  $Y$  (non-trivial means  $\hat{v} \neq 0 \bmod |\mathbb{H}|$ ). Examples of special homomorphisms include Schnorr-like homomorphism<sup>7</sup>  $\psi : \mathbb{Z}_q \rightarrow \mathbb{Z}_p^*$ ,  $\psi : x \mapsto h^x$  with  $\text{ord}(h) = q$ ,  $q \mid (p - 1)$  and Paillier homomorphism<sup>8</sup>.

For special homomorphisms it is sometimes possible to build  $\Sigma$ -protocols with non-binary challenge spaces (and thus small soundness error) by applying Shamir’s trick to just one extracted PP, and the special PP. This is the best known method of extraction for Paillier in the honest setting. However, in the subverted  $N$  scenario it does not work, and binary challenges are still optimal. This is because of the GCD condition in Shamir’s trick:  $\mathcal{A}$  can choose  $N$  to maximize  $\Pr[\gcd(c_1 - c_2, N) \neq 1]$  ( $N$  is a degree of Paillier special PP); with binary challenges  $c_1 - c_2 = 1$ , and GCD is always 1. Other variants of Paillier (e.g. ElGamal-Paillier [15, 31]), are not known to be special, thus even the above extraction technique fails unless challenges are binary ( $c_1 - c_2 = 1$ ).

<sup>7</sup> Its special PP is  $(q, 0)$ , since  $Y^q = \psi(0)$ ; and the PP is non-trivial:  $q \neq 0 \bmod p$ .

<sup>8</sup> From  $Y = G^m r^N$  we can derive  $Y^N = (G^m r^N)^N = G^0 (G^m r^N)^N$ , so  $(N, (0, Y))$  is a pseudo-preimage of degree  $N$  (and  $N \neq 0 \bmod \phi(N^2)$ ).

## 2.4 Designated-Verifier Arguments of Knowledge

We assume some familiarity with the notion of interactive arguments of knowledge and their standard security properties (completeness, knowledge-soundness, and zero-knowledge). In the *designated verifier* (DV) model, additionally to  $\mathcal{P}, \mathcal{V}$  programs we claim existence of a **KeyGen** routine that the verifier uses to create verifier's public key (VPK). This public key is then used to interact with this verifier only, and can potentially be reused multiple times. The formal definitions of completeness, soundness with reusable VPK, and honest verifier zero-knowledge under a malicious VPK are deferred to the full version.

## 3 Our Extraction Technique

In this section we state and prove two lemmas about our novel extraction method. The first is a generalised extraction lemma, Lemma 1, that describes how to extract a witness given  $M$  accepting transcripts such that the gcd of the challenges is 1. Our second lemma, Lemma 2, is the core information-theoretical lemma behind the security of our construction, which argues about this probability of random challenges being coprime.

### 3.1 The Generalized Extraction Lemma

We consider the three-round public-coin protocol of Fig. 1 where transcripts have the form  $(a, c, s)$ . In Lemma 1 we design an extractor that, given  $M$  valid transcripts on the same first message, always succeeds provided that  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) = 1$ . The following is proven in the full version.

**Lemma 1.** *Let  $\mathcal{T} = \{(a, c^{(i)}, s^{(i)})\}_{i=1}^M$  be a collection of  $M \geq 3$  successful transcripts for the relation  $\mathcal{R}_{\text{Hom}}$  and input  $Y$ ,  $aY^{c^{(i)}} = \psi(s^{(i)})$ , such that  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) = 1$ . Then there exists a PPT extractor **Ext** that outputs  $w$  such that  $Y = \psi(w)$  with probability 1.*

### 3.2 Our Core Coprimality Lemma

The above generalized extraction technique is effective conditioned on the fact that differences of the challenges in the extracted transcripts are coprime,  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) = 1$ . However, this cannot be guaranteed for any malicious prover. This stems from the fact that an adversarial prover can manipulate the  $c^{(i)}$ 's by selectively choosing to answer successfully or not, after receiving  $c^{(i)}$ .

Intuitively, we would like the adversary to answer independently of  $c^{(i)}$ . Then for sufficiently large  $M = \text{poly}(\lambda)$ ,  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) = 1$  would hold. To this end we let the challenges consist of two factors: the challenge is  $e = \mathbf{bc}^T$  where  $\mathbf{b}$  is sampled during the protocol execution and  $\mathbf{c}$  is a vector that is uniformly random from the point of view of the adversary. The adversary can manipulate  $\mathbf{b}$  because  $\mathbf{b}$  is chosen during the protocol, but  $\mathbf{c}$  cannot be manipulated. Looking ahead, in Sect. 4 we realize this technique in the designated-verifier setting.

In Lemma 2 we prove an information-theoretical statement which is at the core of our construction. The distribution of values output by our extractor depend nontrivially on some adversarial matrix  $\mathbf{B}$ : the matrix of all  $\mathbf{b}$  that the adversary chooses to answer successfully. Because there are no computational restrictions on how an adversary might choose  $\mathbf{B}$ , we require that for any  $\mathbf{B}$  the extractor will succeed with high probability. Lemma 2 is new to this work and as far as we are aware there are no similar results in the literature.

**How to Interpret the Lemma.** As previously noted, Lemma 2 aims to information-theoretically prove that  $M$  extracted accepting transcripts (on the same first message) have coprime challenges where each challenge is  $\mathbf{b}^{(i)}\mathbf{c}^T$ . From the point of view of the adversary  $\mathbf{b}$  is known but  $\mathbf{c}$  is not, and assumed uniformly random.

To make the applicability of the lemma more clear we briefly recall (omitting the non-relevant details) the extractor of [2] (that generalizes [32]) which obtains  $M$  accepting transcripts, with the same first message, for any  $\Sigma$ -protocol.

Let  $\mathbf{H}$  be the binary matrix where the rows represent the first messages  $\alpha_1 = \psi(r_1), \alpha_2 = \psi(r_2), \dots, \alpha_{|\mathcal{D}|} = \psi(r_{|\mathcal{D}|})$  and the columns represent the different challenges  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{2^\lambda}$ . The position  $\mathbf{H}_{i,j}$  is 1 if the adversary can answer successfully on  $\alpha_i, \mathbf{b}_j$  and 0 otherwise. The extractor works as follows:

- Probes different positions of  $\mathbf{H}$  until it finds a 1.
- If it finds a first 1 it continues sampling uniformly in the same row until it finds  $M - 1$  more 1's (or terminates with some specific probability).

Attema et al. [2] show that this extraction strategy outputs  $M$  accepting transcripts in expected polynomial time.

Assume that the extractor succeeds in outputting the  $M$  transcripts from some row  $i$ . Then  $\mathbf{B}$  (in matrix form) represents all the  $\mathbf{b}_j$ 's of this row that have 1. Similarly,  $\mathbf{B}'$  (also in matrix form) represents all the  $\mathbf{b}^{(j)}$ 's of the row that were sampled (uniformly) by the extractor, contained 1 and thus gave an accepting transcript. Lastly, for the lemma to be applied we need that  $\mathbf{B}$  has exponentially large number of rows  $> 2^\lambda/\text{poly}(\lambda)$ . Conditioned on the fact that the extractor terminates in (expected) polynomial time this holds, otherwise the probability of the extractor to find  $M$  1's in the row (in poly-time) would be negligible. Clearly then,  $\mathbf{B}'$  is a polynomially sized sub-matrix of  $\mathbf{B}$ .

We highlight that the matrix  $\mathbf{H}$  represents the malicious prover's strategy and it is clearly adversarially chosen, thus so is  $\mathbf{B}$ . For this it is important that the lemma holds for any arbitrary  $\mathbf{B}$ . This makes the lemma and its proof highly non-trivial.

**Lemma Statement.** Lemma 2 proves the following. Assume *any* exponentially-large  $(2^\lambda/\text{poly}(\lambda))$  space  $\mathbf{B}$  of binary vectors with  $\lambda$  coordinates. Then if we sample uniformly  $M = \text{poly}(\lambda)$  vectors from this space  $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(M)} \stackrel{\S}{\leftarrow}$



$\mathbf{B}$  and  $\lambda$  uniformly random values (from an exponentially large space)  $\mathbf{c} := (c_1, \dots, c_\lambda) \leftarrow_{\$} ([2^\lambda])^\lambda$  we get that their inner products  $\mathbf{b}^{(1)}\mathbf{c}^T, \dots, \mathbf{b}^{(M)}\mathbf{c}^T$  are coprime, except with negligible probability. This then generalizes to our final result that concerns with the differences  $\{\mathbf{b}^{(i)}\mathbf{c}^T - \mathbf{b}^{(1)}\mathbf{c}^T\}_{i=2}^M$  being coprime.

Crucially, this holds for any space  $\mathbf{B}$  as long as it is sufficiently large.

**Lemma 2.** *Let  $\mathbf{B}$  be any  $(\epsilon'2^\lambda) \times \lambda$  binary matrix consisting of  $\epsilon'2^\lambda$  distinct binary rows, with  $\epsilon' > 1/\text{poly}(\lambda)$ . Sample:*

- $M = \text{poly}(\lambda)$  rows of  $\mathbf{B}$ ,  $i_k \leftarrow_{\$} [1, \epsilon'2^\lambda]$  for  $k = 1, \dots, M$ , and set

$$\mathbf{B}' = (\mathbf{b}^{(1)} \mathbf{b}^{(2)} \dots \mathbf{b}^{(M)})^T := (\mathbf{b}_{i_1} \mathbf{b}_{i_2} \dots \mathbf{b}_{i_M})^T$$

- $\lambda$  uniformly random values,  $c_i \leftarrow_{\$} [2^\lambda]$  for  $i = 1, \dots, \lambda$ , and set

$$\mathbf{c} = (c_1 \ c_2 \ \dots \ c_\lambda)$$

and set  $(e^{(1)} \dots e^{(M)})^T = \mathbf{B}'\mathbf{c}$ . Then:

$$\Pr[\text{gcd}(e^{(2)} - e^{(1)}, \dots, e^{(M)} - e^{(1)}) = 1] = 1 - \text{negl}(\lambda)$$

the probability is over the choices of  $\mathbf{c}, \mathbf{B}'$ .

Due to space limitations the full proof is deferred to full version.

## 4 Designated Verifier Proofs of Knowledge for General Homomorphisms

In this section we design a designated verifier argument of knowledge for an opening to a general homomorphisms. We prove that there is a negligible soundness error assuming an additively homomorphic encryption scheme that is CPA secure. Zero-knowledge holds even under subverted parameters and it does not require a common reference string. Our proofs consist of 6 elements and can be made non-interactive using the Fiat-Shamir transform.

We show in Sect. 5 that knowledge of an opening for a general homomorphism is powerful enough to build range proofs for ciphertexts over a subverted encryption key. For now we focus on the simpler general relation

$$\mathcal{R}_{\text{Hom}} = \{ \psi, A \mid w : Y = \psi(w) \}$$

where  $\psi : \mathcal{D} \rightarrow \mathbb{H}$  and  $\mathbb{H}$  is a group parametrized by a maliciously generated RSA modulus  $N$  (for example  $\mathbb{Z}_N^*$  or  $\mathbb{Z}_{N^2}^*$ ). Although not directly in our scope, the techniques of this sections also apply to any group of unknown order.

## 4.1 The Designated-Verifier Protocol

We are now ready to present our designated verifier zero-knowledge proof system for  $\mathcal{R}_{\text{Hom}}$  where  $\psi$  is any additive group homomorphism.

The public-coin interactive DV protocol for  $\mathcal{R}_{\text{Hom}}$  is run between a prover and the verifier. The protocol is a modification of the sigma protocol in Fig. 1 to ensure soundness even for subverted RSA groups. One of the key observations is that in the Designated-Verifier setting we can hide the challenge from the malicious prover. We can thus assume that *all* the challenges answered are independent, provided that they are sampled independently by the verifier. In order to hide the challenges from the prover they are encrypted with a public key homomorphic encryption scheme. These encrypted challenges are provided in advance inside the verifier's public key.

Then if these encrypted challenges are linearly combined with fresh (binary) challenges, sampled during the actual execution one can directly apply the extraction techniques of Sect. 3 (Lemma 1 and Lemma 2). The linear combination is performed homomorphically through the ciphertexts.

The full protocol is presented in  $\text{DV}_{\text{Prot}}$ . For ease of presentation, we first describe our protocol incrementally: with respect to a trusted setup that always outputs  $(\text{vpk}, \text{vsk})$  honestly and without allowing any reusability of it; then in the next sections we incrementally present how to achieve these properties.

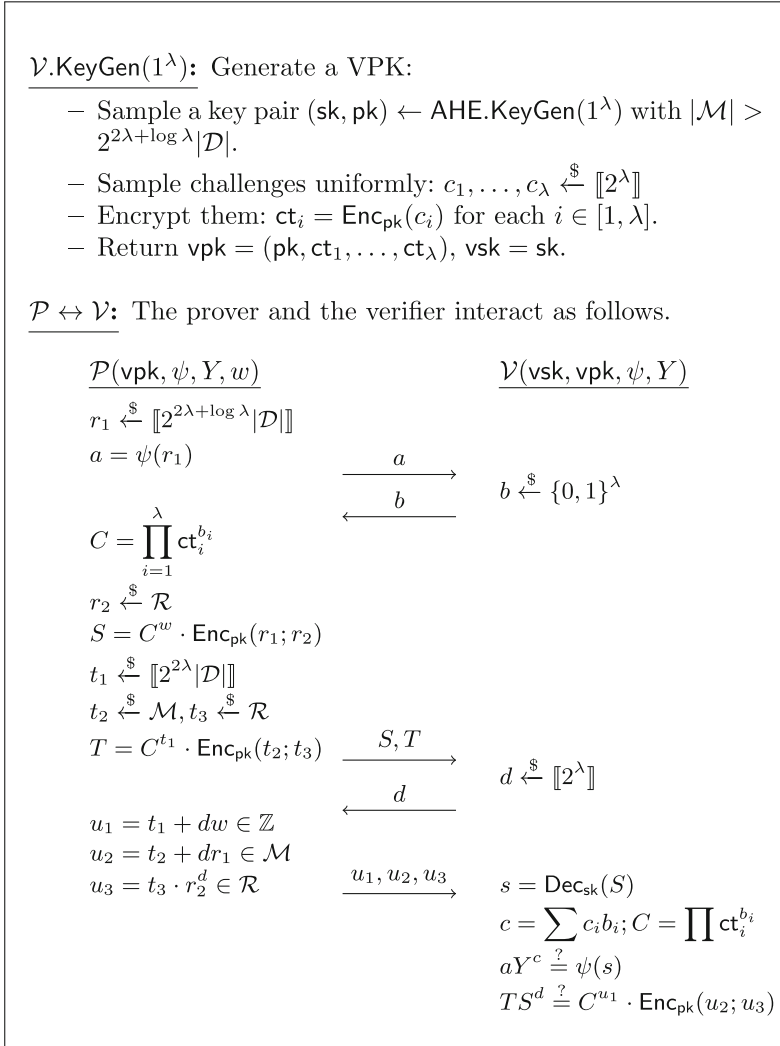
Our construction makes use of any additive additively homomorphic encryption scheme with message space  $\mathcal{M}$ , randomness space  $\mathcal{R}$ , and ciphertext space  $\mathcal{CT}$  such that  $\mathcal{CT}$  forms a multiplicative group. For simplicity we will assume AHE to be standard Paillier w.r.t.  $N_{\text{pk}}$ , and  $\mathcal{M}$  to be the ring  $\mathbb{Z}_{N_{\text{pk}}}$  for an integer  $N_{\text{pk}}$ , although our scheme works with any AHE and ring  $\mathcal{M}$ .<sup>9</sup>

First the key generation algorithm creates a verification key: it chooses an encryption key pair  $(\text{pk}, \text{sk})$  and sets the verifier's secret key to  $\text{vsk} = \text{sk}$ . It then samples uniformly  $\lambda$  values,  $c_1, \dots, c_\lambda \stackrel{\$}{\leftarrow} \llbracket 2^\lambda \rrbracket$  (denote  $\mathbf{c} = (c_1, \dots, c_\lambda)$ ) and encrypts them under  $\text{pk}$ ,  $\text{ct}_1 = \text{Enc}_{\text{pk}}(c_1), \dots, \text{ct}_\lambda = \text{Enc}_{\text{pk}}(c_\lambda)$ . In Sect. 4.3 we describe a protocol by which the verifier proves that their  $\text{vpk}$  is well formed, ensuring that we achieve zero-knowledge under subverted  $\text{vpk}$  (hence without trusting the designated verifier for the key setup).

The protocol then proceeds in 5 moves which we detail in Fig. 2. The prover essentially proves that  $Y = \psi(w)$  by sending  $a = \psi(r)$ ; an encryption  $S$  of  $(r + cw)$ ; and a proof  $(T, u_1, u_2, u_3)$  that the prover knows the contents of  $S$ . The additional steps 4 and 5 that prove knowledge of the preimage of  $S$  are there so that we can technically avoid passing  $\text{vsk}$  to the extractor to compute  $s$ . Instead they can extract  $s$  from the additional protocol of these steps. This explains why  $d$  is sampled from the exponentially big challenge space – the modulus in question (chosen by the verifier and extractor) is trusted for soundness.

As usual in public-coin protocols, the interactive  $\text{DV}_{\text{Prot}}$  can be transformed into a non-interactive one applying the Fiat-Shamir transformation (in the random oracle model).

<sup>9</sup> As long as all elements in  $\llbracket 2^{\lambda+1} \rrbracket$  have a multiplicative inverse in  $\mathcal{M}$ .



**Fig. 2.**  $\text{DV}_{\text{Prot}}$ : The designated-verifier  $\Sigma$ -protocol for  $\mathcal{R}_{\text{Hom}}$  demonstrating knowledge of a preimage of  $\psi(\cdot)$ . The additively homomorphic encryption scheme is instantiated with Paillier with  $|\mathcal{M}| = |N_{\text{pk}}|$ . This scheme is knowledge sound for subverted RSA groups provided that the outputs of  $\text{KeyGen}(1^\lambda)$  are well-formed.

### 4.2 Security

We now argue the security of our  $\text{DV}_{\text{Prot}}$ . For correctness, we only need to make sure that the message space  $\mathcal{M}$  of AHE is large enough to fit the largest possible  $s = r_1 + cw$ . That is we require an additively homomorphic IND-CPA secure Encryption Scheme with message space  $|\mathcal{M}| > 2^{2\lambda + \log \lambda} |\mathcal{D}|$ .

**Knowledge Soundness.** To demonstrate knowledge soundness we first describe an extractor that can rewind a malicious prover and aims to output the prover’s witness. This extractor obtains  $M(\lambda) = \text{poly}(\lambda)$  different verifying transcripts from the prover and succeeds if the gcd of the challenges of these transcripts is equal to 1. We then describe a reduction  $\mathcal{B}$  that succeeds at IND-CPA whenever the extractor fails at obtaining a valid witness. The reduction queries an encryption oracle to determine the  $\text{vpk}$  and therefore does not know the contents of the encryptions. It runs the prover and decides whether a transcript verifies or not based on whether the transcript verifies with *both* possible contents. We argue that if it verifies with one of the possible contents but not the other, then provided the domain space of  $\psi(\cdot)$  is bigger than  $2^\lambda$ , then  $\mathcal{B}$  can guess the contents of the ciphertexts with overwhelming probability. We further argue that the gcd of the challenges the prover does not see must equal 1 with overwhelming probability. Thus if the extractor fails then  $\mathcal{B}$  can guess which challenges the ciphertexts contain based on whether the gcd is 1 or not.

The protocol and theorem currently do not give the prover oracle access to the verifier. In Sect. 4.4 we will describe an extension of our DV protocol that can give the prover this access.

**Theorem 1 (Knowledge Soundness).** *The  $\text{DV}_{\text{Prot}}$  protocol is knowledge-sound in the designated verifier model, provided that the AHE is IND-CPA secure.<sup>10</sup>*

*Proof.* Suppose that  $(\text{vpk}, \text{vsk}, \tau) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$ , where  $\tau = \{c_1, \dots, c_\lambda\}$  contains the challenges encrypted in  $\text{vpk}$  but not the secret key  $\text{sk}$  of AHE. Assume that  $\mathcal{P}^*(\text{vpk}, \psi, Y; \text{coin})$  is a malicious prover that is run on random coins  $\text{coin}$ . We first describe an extractor  $\text{Ext}$ , that has rewindable black-box access to the prover  $\mathcal{P}^*$ , such that whenever  $\mathcal{P}^*$  outputs verifying  $(Y; (a, S, T, u_1, u_2, u_3))$   $\text{Ext}^{\mathcal{P}^*}(\tau, \text{vpk}, \psi, Y)$  outputs a witness  $w$  such that  $Y = \psi(w)$ . The  $\text{Ext}$  algorithm depends on two subalgorithms,  $\text{Ext}_0$  and  $\text{Ext}_1$  where  $\text{Ext}_0$  is the extractor from Lemma 1, and  $\text{Ext}_1$  we present below.

$\text{Ext}_1$ , on input  $\tau, \text{vpk}, \psi$  and  $Y$ , runs  $\mathcal{P}^*(\text{vpk}, \psi, Y; \text{coin})$  (on challenges  $b, d$  of its choice) until it obtains a full  $(M, 2)$ -tree of accepting transcripts, for the same first message  $a$ . That is:

$$\mathcal{T} = \left\{ \left( a, b^{(j)}, S^{(j)}, T^{(j)}, d^{(j,k)}, u_1^{(j,k)}, u_2^{(j,k)}, u_3^{(j,k)} \right) \right\}_{j \in [M], k \in [2]}$$

and outputs  $\mathcal{T}$ . For  $\text{Ext}_1$  we use the generic  $(M, 2)$ -special soundness extractor (see [2]), that efficiently finds such a tree. As we argue later we set  $M = \text{poly}(\lambda)$ .

More specifically,  $\text{Ext}_1$  proceeds as follows. It probes  $\mathcal{P}^*$  on randomly sampled  $\text{coin}, b, d$  until it obtains  $\left( a, b^{(1)}, S^{(1)}, T^{(1)}, d^{(1,1)}, u_1^{(1,1)}, u_2^{(1,1)}, u_3^{(1,1)} \right)$  such that  $T^{(1)}(S^{(1)})^{d^{(1,1)}} = (C^{(1)})^{u_1^{(1,1)}} \text{Enc}_{\text{pk}}(u_2^{(1,1)}; u_3^{(1,1)})$ , where  $C^{(1)} = \prod_{i=1}^\lambda \text{ct}_i^{b_i^{(1)}}$ . Since it does not have  $\text{vsk}$  it cannot directly decrypt  $S^{(1)}$  to  $s^{(1)}$  and check whether

<sup>10</sup> We further assume that if  $\mathbb{Z}_N$  is the message space, then the largest factor of  $N$  is larger than  $2^{\lambda+1}$ , which is the case for example in Paillier.

$aY^{c^{(1)}} = \psi(s^{(1)})$ . For this it continues probing  $\mathcal{P}^*$  on the same coin and  $b^{(1)}$  until it obtains a second  $(a, b^{(1)}, S^{(1)}, T^{(1)}, d^{(1,2)}, u_1^{(1,2)}, u_2^{(1,2)}, u_3^{(1,2)})$  such that  $T^{(1)}(S^{(1)})^{d^{(1,2)}} = (C^{(1)})^{u_1^{(1,2)}} \text{Enc}_{\text{pk}}(u_2^{(1,2)}; u_3^{(1,2)})$ . So we have:

$$\begin{aligned} T^{(1)}(S^{(1)})^{d^{(1,1)}} &= (C^{(1)})^{u_1^{(1,1)}} \text{Enc}_{\text{pk}}(u_2^{(1,1)}; u_3^{(1,1)}) \\ T^{(1)}(S^{(1)})^{d^{(1,2)}} &= (C^{(1)})^{u_1^{(1,2)}} \text{Enc}_{\text{pk}}(u_2^{(1,2)}; u_3^{(1,2)}) \end{aligned}$$

or

$$(S^{(1)})^{d^{(1,1)} - d^{(1,2)}} = \text{Enc}_{\text{pk}}(u_2^{(1,1)} + c^{(1)}u_1^{(1,1)} - u_2^{(1,2)} - c^{(1)}u_1^{(1,2)})$$

From assumption  $\gcd(d^{(1,1)} - d^{(1,2)}, N) = 1$  (given that the largest prime factor of  $N$  is larger than  $|d^{(1,1)} - d^{(1,2)}|$ ) so the inverse  $(d^{(1,1)} - d^{(1,2)})^{-1}$  exists in  $\mathcal{M}$  and  $\text{Ext}_1$  extracts  $s^{(1)} = s_2^{(1)} + c^{(1)}s_1^{(1)}$  such that  $S^{(1)}$  encrypts  $s^{(1)}$  (under some randomness unknown to the extractor) where

$$\begin{aligned} s_1^{(1)} &= (u_1^{(1,1)} - u_1^{(1,2)}) (d^{(1,1)} - d^{(1,2)})^{-1} \pmod N \\ s_2^{(1)} &= (u_2^{(1,1)} - u_2^{(1,2)}) (d^{(1,1)} - d^{(1,2)})^{-1} \pmod N \end{aligned}$$

From here  $\text{Ext}_1$  can verify  $aY^{c^{(1)}} = \psi(s^{(1)})$  to confirm if the two transcripts are accepting or not. It continues in a similar manner until it obtains a full  $(M, 2)$ -tree of accepting transcripts  $\mathcal{T}$ . Whenever  $\mathcal{P}^*$  convinces  $\mathbb{V}$  with non-negligible probability  $\text{Ext}_1$  computes the decryption of  $S^{(1)}$  in polynomial time thus the probability that  $\text{Ext}_1$  accepts a false transcript is negligible.<sup>11</sup>

Now, the extractor  $\text{Ext}$  behaves as follows. It runs  $\mathcal{T} \leftarrow \text{Ext}_1^{\mathcal{P}^*}(\tau, \text{vpk}, \psi, Y)$  and computes  $c^{(j)} = \mathbf{b}^{(j)} c^T = \sum_{i=1}^\lambda c_i b_i^{(j)}$ . If  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(\lambda)} - c^{(1)}) \neq 1$  it aborts. Else it computes  $s^{(j)}$  as shown above (where it holds that  $s^{(j)} = \text{Dec}_{\text{sk}}(S^{(j)})$ ) for each  $j \in [M]$  and runs  $w \leftarrow \text{Ext}_0(\psi, Y; (a, c^{(1)}, s^{(1)}), \dots, (a, c^{(M)}, s^{(M)}))$  and returns  $w$ .

We first see that  $\text{Ext}$  runs in polynomial time provided that the adversary  $\mathcal{P}^*$  has non-negligible probability of success. So either  $\epsilon(\lambda)$  is polynomial in  $\lambda$  or  $\mathcal{P}^*$  only convinces  $\mathbb{V}$  with negligible probability. Let  $\epsilon(\lambda) > 1/\text{poly}(\lambda)$  denote the probability that  $\mathcal{P}^*$  convinces an honest verifier on input  $(\psi, Y)$ . By Lemma 1 we have that  $\text{Ext}_0$  runs in polynomial time. For the runtime of  $\text{Ext}_1$  we rely on [2, Lemma 5] which shows that  $\text{Ext}_1$  runs in expected time  $O(\frac{\lambda}{\epsilon - (M-1)/2^\lambda})$ , which is polynomial (since we assumed that  $\epsilon$  is non-negligible).

We must now show that  $\text{Ext}$  only aborts with negligible probability. This occurs if and only if  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) \neq 1$  with non-negligible probability. In order to show this, we design an adversary  $\mathcal{B}$  against IND-CPA that, using  $\text{Ext}$ , wins the IND-CPA game:

<sup>11</sup> For ease of exposition we keep the description simple. We omit the technical details of special soundness extractors related to aborting scenarios, that ensure termination in polynomial time (see lemma 5, [2]).

$\mathcal{B}^{\mathcal{O}_{\text{Enc}}}(\text{pk})$   
 $c_1, z_1, \dots, c_\lambda, z_\lambda \xleftarrow{\$} \llbracket 2^\lambda \rrbracket$   
 $\text{ct}_i \xleftarrow{\$} \mathcal{O}_{\text{Enc}}(c_i, z_i) \quad \text{for } i \in [\lambda];$   
 $\text{vpk} \leftarrow (\text{pk}, \text{ct}_1, \dots, \text{ct}_\lambda)$   
 $\text{coin} \xleftarrow{\$} [1, 2^\lambda]; j \leftarrow 1$   
 while  $j < M$  :  $(\text{trans}_{j,1}, \text{trans}_{j,2}) \leftarrow \mathcal{P}^*(\text{vpk}, \psi, Y; \text{coin})$   
     if  $aY^{c^{(j)}} = \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  and  $aY^{z^{(j)}} \neq \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$  return 0  
     if  $aY^{c^{(j)}} \neq \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  and  $aY^{z^{(j)}} = \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$  return 1  
     if  $aY^{c^{(j)}} = \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  and  $aY^{z^{(j)}} = \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$   $j \leftarrow j + 1$   
 if  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) \neq 1$  return 0  
 if  $\gcd(z^{(2)} - z^{(1)}, \dots, z^{(M)} - z^{(1)}) \neq 1$  return 1

where we denote  $c^{(j)} = \mathbf{b}^{(j)} \mathbf{c}^T$  and  $z^{(j)} = \mathbf{b}^{(j)} \mathbf{z}^T$ .

*Case 1.* First we show that if  $aY^{c^{(j)}} = \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  and  $aY^{z^{(j)}} \neq \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$ , then with overwhelming probability the encryptions contain  $c_1, \dots, c_\lambda$  and  $\mathcal{B}$  succeeds.

The fact that  $aY^{c^{(j)}} = \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  can be rewritten as:

$$\left( a\psi(-s_2^{(j)}) \right) = \left( \psi(s_1^{(j)})Y^{-1} \right)^{c^{(j)}}$$

Assume that  $\text{ct}_i \neq \text{Enc}_{\text{pk}}(c_i)$  then  $\mathcal{P}^*$  gets no information about  $c_1, \dots, c_\lambda$ , so they are perfectly hidden. This means that from the point of view of  $\mathcal{P}^*$  these are uniformly random over  $\llbracket 2^\lambda \rrbracket$ , which makes the above happen with probability  $2^{-\lambda}$  (considering also that  $|\mathbb{H}| > 2^\lambda$ ), unless  $a\psi(-s_2^{(j)}) = \psi(s_1^{(j)})Y^{-1} = 1$ . Now, since  $aY^{z^{(j)}} \neq \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$  then  $a \neq \psi(s_2^{(j)})$  or  $Y \neq \psi(s_1^{(j)})$ .

We conclude then that, except with negligible probability  $2^{-\lambda}$ ,  $\{\text{ct}_i\}_i$  contain encryptions of  $c_i$ .

*Case 2.* Second, we use the same argument as in the previous case to claim that if  $aY^{c^{(j)}} \neq \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  and  $aY^{z^{(j)}} = \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$ , then with overwhelming probability the encryptions contain  $z_1, \dots, z_\lambda$  and  $\mathcal{B}$  succeeds.

*Case 3.* Third we argue that if the extractor Ext fails then  $\mathcal{B}$  succeeds. Indeed we have from the first two cases that transcripts only verify if both  $aY^{c^{(j)}} = \psi(s_2^{(j)} + c^{(j)}s_1^{(j)})$  and  $aY^{z^{(j)}} = \psi(s_2^{(j)} + z^{(j)}s_1^{(j)})$ . If the encryptions contain  $c_1, \dots, c_\lambda$  then Ext only fails if  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) \neq 1$ . In this case  $\mathcal{B}$  correctly guesses.

If the encryptions instead contain  $z_1, \dots, z_\lambda$  then Ext only fails if  $\gcd(z^{(2)} - z^{(1)}, \dots, z^{(M)} - z^{(1)}) \neq 1$ . In this case  $\mathcal{B}$  guesses correctly unless  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) \neq 1$ . The  $(c_1, \dots, c_\lambda)$  are uniformly distributed values that are perfectly hidden from the prover and the extractor. Indeed, the encryptions contain no information and, by the first two cases, the behaviour of the extractor

is entirely determined by the verification with respect to  $z_1, \dots, z_\lambda$ . So the probability that  $\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) = 1$  is overwhelming (see Lemma 2). We thus argue that if Ext fails then  $\mathcal{B}$  succeeds with overwhelming probability.

Indeed Lemma 2 shows that  $\Pr[\gcd(c^{(2)} - c^{(1)}, \dots, c^{(M)} - c^{(1)}) = 1] = 1 - \text{negl}(\lambda)$ .

To see why Lemma 2 applies in our case,  $\mathbf{B}$  corresponds to the matrix containing all the challenges  $b$  which the adversary can successfully answer, when the first message is  $a$ . Since the extractor was able to obtain  $M$  such challenges in (expected) polynomial time, this means that  $\mathbf{B}$  is at most polynomially smaller than  $2^\lambda$ : there exists  $\epsilon' > 1/\text{poly}(\lambda)$  such that  $|\mathbf{B}| = \epsilon' 2^\lambda$ . We can show this by contradiction, assume that  $\epsilon' = 1/\omega(\text{poly}(\lambda))$ , then the expected time for Ext to find a successful answer would be non-polynomial  $\omega(\text{poly}(\lambda))$ . Finally,  $\mathbf{B}'$  corresponds to the matrix consisting of the challenges in  $\mathcal{T}$ .

**Zero-Knowledge.** To demonstrate zero-knowledge we will provide a simulator and argue that the simulator's outputs are indistinguishable from the honest provers. We make use of a standard blinding lemma.

The main HVZK result is as follows (due to space limitations the proof is deferred to the full version of the paper):

**Theorem 2 (Honest Verifier Zero Knowledge).**  $\text{DV}_{\text{Prot}}$  is statistical honest-verifier zero-knowledge for the relation  $\mathcal{R}_{\text{Hom}}$ .

Since our DV protocol is essentially Schnorr-like, the simulator is almost as usual: it samples response values uniformly (since they are properly blinded in the honest protocol), and generates (encrypted) challenges using verifier's equations. The only difference is that one challenge is an encryption value. Also the proof assumes honest CRS setup.

### 4.3 Malicious VPK Generation

The  $\text{DV}_{\text{Prot}}$  protocol in the previous section assumes that the verifier's public key is trusted. In particular, zero-knowledge only holds on the condition that  $\text{ct}_i$  contains plaintexts  $c_i \in \llbracket 2^\lambda \rrbracket$  for all  $i$ . In this section we explain how to generate a  $\text{vpk}$  in a way that prevents dishonest verifiers from breaking zero-knowledge of our DV construction.

For lack of space we defer the formal description of the malicious-verifier alternative key generation procedure to the full version. We edit the setup algorithm such that the verifier must provide a range proof on the ciphertexts it generates for  $\text{vpk}$ .

In the full version we present a protocol proving range of the VPK ciphertext efficiently, together with a security proof. The protocol follows the transformation by Cramer et al. [29,30] allowing to increase performance when proving multiple instances simultaneously; however our instantiation has a number of differences from the original transformation. The range proof comes with a slack:

a verifying  $\pi$  on the prover's side guarantees that when  $c_i \in \llbracket 2^\lambda \rrbracket$ , the resulting messages in the ciphertexts  $\text{ct}_i$  of  $\text{vpk}$  are in the extended interval  $\llbracket 2^{3\lambda + \log \lambda - 1} \rrbracket$  (the slack is  $2^{2\lambda + \log \lambda - 1}$ ). Therefore the encrypted sum-challenge  $\mathcal{P}$  replies to is in  $\llbracket 2^{3\lambda + 2 \log \lambda - 1} \rrbracket$ . To preserve zero-knowledge we must increase the blinding parameter  $r_1$  on the prover's side to this value, multiplied by  $|\mathcal{D}|$ . This in turn requires us to increase AHE  $|\mathcal{M}|$  to  $|\mathcal{D}|2^{3\lambda + 2 \log \lambda}$ , to be enough to fit the new  $s = r_1 + cw \stackrel{\approx}{\approx} r_1$ .

In addition to this, we also must prove that verifier's public key  $N_{\text{pk}}$  gives rise to an *injective* Paillier instantiation, since otherwise the statement of the range proof is not useful. For this we use [42, Protocol  $\mathcal{P}_{\text{Paillier-N}}$ , Sect. 3.2]—it is public-coin, so can be executed non-interactively (using FS); it proves  $\text{gcd}(N_{\text{pk}}, \phi(N_{\text{pk}})) = 1$ , which is enough to achieve injectivity of Paillier; and it is quite efficient, only taking a few percent of all KeyGen computations.

#### 4.4 Reusable VPK

In this section we present  $\text{DVReusable}_{\text{Prot}}$ , a modification of  $\text{DV}_{\text{Prot}}$ , in which  $\text{vpk}$  is reusable  $Q = \text{poly}(\lambda)$  number of times. This means the prover can query the verifier to learn whether their response verifies up to  $Q$  times. We achieve this by adding  $Q$  encrypted challenges to the  $\text{vpk}$ . The result is that both the communication and the computation complexity related to  $\text{vpk}$  generation and verification can be amortized down to  $O(1)$  per query.

For the basic  $\text{DV}_{\text{Prot}}$  it is possible to show an adversarial prover, interacting with the verifier many times, uses the information of whether a (malicious) proof of their choice verifies or not in order to learn plaintext challenges  $c_i$  in the  $\text{vpk}$ . This in turn defeats the purpose of hiding the challenges, and prevents extraction, breaking soundness.

To overcome this we introduce additional challenge blinders. First, we sample  $\hat{c}_\kappa$  of size at least  $\lambda 2^{2\lambda}$  per query, encrypt them to  $\hat{\text{ct}}_\kappa$ , and add them all to the VPK. Then we use  $\hat{\text{ct}}_\kappa$  in the final challenge  $C = \hat{\text{ct}}_\kappa \prod_i \text{ct}_i^{b_i}$  (for a challenge bit-vector  $b$ ) so that  $\hat{c}_\kappa$  statistically hides  $\sum c_i b_i$  since  $\hat{c}_\kappa$  is at least  $2^\lambda$  larger. This means that the adversary statistically learns no information about  $\{c_i\}$ , but only about  $\hat{c}_\kappa$ . Each challenge  $\hat{c}_\kappa$  must be used exactly once, which is enforced by  $V$ .

The final challenge size now grows to  $\lambda 2^{2\lambda}$ , which means  $r_1$  must be sampled from  $\llbracket \lambda 2^{3\lambda} |\mathcal{D}| \rrbracket$ , and  $|\mathcal{M}|$  of verifier's AHE must be bigger than this value.

**Theorem 3.**  *$\text{DVReusable}_{\text{Prot}}$  is a complete, honest-verifier zero-knowledge protocol in the designated-verifier setting, that has knowledge-soundness with  $Q$ -times reusable VPK for any polynomial  $Q(\lambda)$ .*

Due to space limitations the proof is deferred to the full version.

#### 4.5 Malicious and Reusable VPK

Techniques from the two previous sections can be combined. The *reusable* VPK from Sect. 4.4 can also be generated *maliciously* with the same technique from Sect. 4.3.



The batched range proof now must also cover new “bigger” challenges introduced for reusability. From the perspective of efficiency of amortized  $\text{SigmaRangeA}_{\text{Prot}}$  it is optimal to batch exactly  $n = \lambda$  instances together. Thus we will prove challenge ranges of  $c_i$  in batches of size  $\lambda$ , where first batch uses range bound  $R_1 = 2^\lambda$  (corresponding to small ciphertexts), and the following  $Q/\lambda$  batches use  $R_2 = \lambda 2^{2\lambda}$ . When  $\lambda \nmid Q$ ,  $\text{SigmaRangeA}_{\text{Prot}}$  instance can be padded with dummy values.

Given  $2^{\lambda + \log \lambda - 1}$  slack of the range proof, we must sample  $r_1 \in \llbracket 2^{5\lambda + 2 \log \lambda} \mathcal{D} \rrbracket$ ; and  $|\mathcal{M}|$  must be chosen to be bigger than this  $r_1$ .

### 4.6 Efficiency Optimization in the Generic Group Model

Here we describe a variant of the  $\text{DV}_{\text{Prot}}$  protocol that consists of 3 rounds (instead of 5) and thus saves 4 elements from the proof size. The protocol transcript simply consists of  $(a, b, S)$  omitting  $T, d, u_1, u_2, u_3$  together with the last two rounds.

In  $\text{DV}_{\text{Prot}}$  the last three messages  $T, d$  and  $(u_1, u_2, u_3)$  are used to prove that  $S$  is a well-formed ciphertext. Namely, the extractor of Theorem 1, at each accepting transcript should be able to obtain an  $s^{(j)}$  such that  $S^{(i)} = \text{Enc}_{\text{pk}}(s^{(j)})$ . We observe that if we instantiate the encryption scheme with the Paillier-with-randomness-in-the-exponent cryptosystem,  $S^{(j)} = (N + 1)^{s^{(j)}} h^r$  then our extractor can obtain  $s^{(j)}$  for free in the generic group model [52, 59] (GGM).

GGM for unknown order groups has been established [13, 37] in a similar manner to the original model. For this optimization we make use of this model. For knowledge-soundness we assume that the group generated for the Paillier encryption is honest (it’s part of VPK), thus the model applies normally.

The following proof is almost identical to that of Theorem 1 except that the extractor now uses whitebox access to the prover instead of the rewinding argument to find a representation for  $S$ .

**Theorem 4 (Knowledge Soundness).** *The optimised  $\text{DV}_{\text{Prot}}$  described above is knowledge-sound in the generic group model provided that the AHE is IND-CPA secure.*

Due to space limitations the proof is deferred to the full version.

## 5 Designated Verifier Range Proof

In this section we construct  $\text{DVRange}_{\text{Prot}}$ —a zero-knowledge argument of knowledge for the range of the pre-image of general homomorphisms. Formally, we are interested in the relation:

$$\mathcal{R}_{\text{HomRange}} = \{(\psi, Y, R); x : Y = \psi(x) \wedge x \in [0, R]\}$$

where  $\psi : \mathcal{D} \rightarrow \mathbb{G}$  and  $\mathbb{G}$  is a group parameterised by a (possibly subverted) RSA modulus  $N$ . We use our designated-verifier protocol of Sect. 4, that is able

to extract the witness using the extraction strategy of Theorem 1. On top of that, we use the range proof from [27] for RSA groups.

The protocol from [27] works over an integer commitment [34,40]  $c = g^x h^r$  in an RSA group for which the order is unknown to the prover. Since we cannot assume that  $\mathbb{G}$  is such a group (recall that the prover might know the order of  $\mathbb{G}$ ) we let the verifier generate an RSA modulus  $N_{\text{cm}}$  together with the bases of the commitment  $g, h$ , which are included in the verification key. The prover first commits to the pre-image  $x$  in  $\mathbb{Z}_{N_{\text{cm}}}$ ,  $c = g^x h^r$  and sends  $c$  to the verifier. Then it performs the two protocols, the opening of  $\psi$  (Sect. 4.1) and the range proof of [27] (compiled with the same Designated-Verifier technique), in parallel.

For completeness, we recall the aforementioned integer commitment scheme used. It works over any group of unknown (to the committer) order such as an RSA group or a class group. In our case, we focus on the RSA instantiation, thus the underlying group is  $\mathbb{Z}_{N_{\text{cm}}}$ , where  $N_{\text{cm}}$  is an RSA modulus. The commitment key consists of two random elements  $g, h \in \mathbb{Z}_{N_{\text{cm}}}$  such that  $g \in \langle h \rangle$ . In the key generation phase we sample uniformly  $g \leftarrow_{\$} \mathbb{Z}_{N_{\text{cm}}}$  and  $f \leftarrow_{\$} \phi(N_{\text{cm}})$ <sup>12</sup> and output  $(g, h) = (h^f, h)$ . A commitment to  $x$  is merely  $c = g^x h^r$  for a random  $r \leftarrow_{\$} \llbracket \frac{N_{\text{cm}}}{2} \rrbracket$ . The opening values are  $(x, r)$  and the verification is  $c = \pm g^x h^r$ .<sup>13</sup> The scheme is binding under the factoring assumption for  $N_{\text{cm}}$  and statistically hiding.

We present  $\text{DVR}_{\text{RangeProt}}$  in Fig. 3 (for lack of space we describe its key generation in the full version). For ease of presentation parts related to the range proof and the opening of  $\psi$  are visually separated, denoted as (1) and (2) respectively. We directly present our protocol with reusable and maliciously generated  $\text{vpk}$ , similarly to how these were presented for  $\text{DV}_{\text{Prot}}$  in Sects. 4.3 and 4.4.

For the key generation, except for a secret/public key of the additively homomorphic encryption scheme (Paillier cryptosystem), we further need an RSA modulus  $N_{\text{cm}}$  and the group elements  $g, h$  to instantiate the integer commitment scheme. For zero-knowledge to hold even under maliciously generated  $\text{vpk}$  it is important that  $g = h^f$  holds. Therefore we additionally include a zero-knowledge proof ensuring it.

**Security.** The above protocol consists of two sub-protocols: our protocol of Sect. 4.1 and the range proof by Couteau et al. [27] over RSA groups. Thus the security of the protocol can be proven in a straightforward way from the security of these subprotocols. For correctness, again we need to consider the size of the message space  $\mathcal{M}$  of the encryption scheme AHE. Indeed  $|\mathcal{M}|$  needs to be at least as large as the maximum value encrypted, which equals  $\tau + \sum_{i=1}^3 x_i t_i - 4(R-x)t$ , the content of  $U_4$ . Knowledge-Soundness follows directly from the knowledge-soundness of the two sub-protocols.

<sup>12</sup> In case  $\phi(N_{\text{cm}})$  is unknown, sampling  $f \leftarrow_{\$} \llbracket \frac{N_{\text{cm}}}{2} \rrbracket$  is statistically close.

<sup>13</sup> The  $\pm$  relaxation is artificially added in order to achieve a sound zero-knowledge proof of opening of  $c$ , which however does not affect the binding of the commitment scheme.

$\mathcal{P}(\text{vpk}, \psi, Y, R, \kappa, x) \leftrightarrow \mathcal{V}(\text{vsk}, \text{vpk}, \psi, Y, R, \kappa)$ :

- $\mathcal{P}_1$ :
1. Sample  $t \leftarrow_{\$} \llbracket 2^\lambda \frac{N_{\text{cm}}}{2} \rrbracket$  and compute  $\text{cm} = g^x h^t \pmod{N_{\text{cm}}}$ .
  2. Sample  $r \leftarrow_{\$} \llbracket 2^{5\lambda+2 \log \lambda} R \rrbracket, \sigma \leftarrow_{\$} \llbracket 2^{6\lambda+2 \log \lambda} \frac{N_{\text{cm}}}{2} \rrbracket$  and compute  $\beta = g^r h^\sigma$ .
  3. Find  $x_1, x_2, x_3 \in \mathbb{Z}$  such that  $4x(R-x) + 1 = \sum_{i=1}^3 x_i^2$  (using e.g. [61]).
  4. Sample  $t_i \leftarrow_{\$} \llbracket 2^\lambda \frac{N_{\text{cm}}}{2} \rrbracket$  and compute  $\text{cm}_i = g^{x_i} h^{t_i}$ , for  $i \in [1, 3]$ .
  5. Sample  $r_i \leftarrow_{\$} \llbracket 2^{5\lambda+2 \log \lambda} R \rrbracket, \sigma_i \leftarrow_{\$} \llbracket 2^{6\lambda+2 \log \lambda} \frac{N_{\text{cm}}}{2} \rrbracket$  and compute  $\beta_i = g^{r_i} h^{\sigma_i}$ , for  $i \in [1, 3]$ .
  6. Sample  $\tau \leftarrow_{\$} \llbracket 2^{6\lambda+2 \log \lambda+4} \frac{N_{\text{cm}}}{2} R \rrbracket$  and compute  $\beta_4 = h^\tau \text{cm}^{4r} \prod_{i=1}^3 \text{cm}_i^{-r_i}$ .
  7. Compute  $\alpha = \psi(r)$ .

$\mathcal{P} \rightarrow \mathcal{V}$ : send  $a = (\text{cm}, \{\text{cm}_i\}_{i \in [1,3]}, \alpha, \beta, \{\beta_i\}_{i \in [1,4]})$

$\mathcal{V}_1$ : Sample  $b \leftarrow_{\$} \{0, 1\}^\lambda$  (denote  $(b_1, \dots, b_\lambda) := b$ ).

$\mathcal{V} \rightarrow \mathcal{P}$ : send  $b$

- $\mathcal{P}_2$ :
1. Compute challenge ciphertext  $C = \text{ct}_{\lambda+\kappa} \cdot \prod_{i=1}^\lambda \text{ct}_i^{b_i}$
  2. Compute:
    - $U = \text{Enc}_{\text{pk}}(r) \cdot C^{R-x}, V = \text{Enc}_{\text{pk}}(\sigma) \cdot C^{-t}$ .
    - $U_i = \text{Enc}_{\text{pk}}(r_i) \cdot C^{x_i}, V_i = \text{Enc}_{\text{pk}}(\sigma_i) \cdot C^{t_i}$ , for  $i \in [1, 3]$ .
    - $U_4 = \text{Enc}_{\text{pk}}(\tau) \cdot C^{\sum_{i=1}^3 x_i t_i - 4(R-x)t}$ .

$\mathcal{P} \rightarrow \mathcal{V}$ : send  $S = (U, V, \{U_i\}_{i \in [1,3]}, \{V_i\}_{i \in [1,3]}, U_4)$

- $\mathcal{V}_2$ :
1. Compute plaintext challenge  $c = c_{\lambda+\kappa} + \sum_{i=1}^\lambda c_i b_i$
  2. Decrypt  $U, V, \{U_i\}_{i \in [1,3]}, \{V_i\}_{i \in [1,3]}, U_4$ :  $u = \text{Dec}_{\text{sk}}(U), v = \text{Dec}_{\text{sk}}(V), u_i = \text{Dec}_{\text{sk}}(U_i), v_i = \text{Dec}_{\text{sk}}(V_i)$  for  $i \in [1, 3]$  and  $u_4 = \text{Dec}_{\text{sk}}(U_4)$
  3. Perform the following checks:
    - $\beta(\text{cm}^{-1} g^R)^c \stackrel{?}{=} g^u h^v$
    - $\beta_i \text{cm}_i^c \stackrel{?}{=} g^{u_i} h^{v_i}$ , for  $i \in [1, 3]$
    - $\beta_4 \prod_{i \in [1,3]} \text{cm}_i^{u_i} \stackrel{?}{=} h^{u_4} g^c \text{cm}^{4u}$
    - $u_i \in \llbracket 2^{5\lambda+2 \log \lambda} R \rrbracket$ , for  $i \in [1, 3]$
    - $\alpha (Y^{-1} \psi(R))^c \stackrel{?}{=} \psi(u)$

$\mathcal{P} \leftrightarrow \mathcal{V}$ : (Non-GGM part:) For each ciphertext of the third message  $S$  perform a variant of the three-round  $\text{Sigma}_{\text{Prot}}$  for the relation  $\mathcal{R} = \{(S_i, C); (w_1, w_2, w_3) : S_i = \text{Enc}_{\text{pk}}(w_1; w_2) \cdot C^{w_3}\}$  with  $|\mathcal{C}| = 2^\lambda$ . This can be done in two extra rounds starting with  $\mathcal{P}_2$ , as in Fig. 2.

**Fig. 3.**  $\text{DVRange}_{\text{Prot}}$ : The designated-verifier range proof of a preimage of  $\psi$ .

**Table 1.** Evaluation of our main protocols. Timings are in ms. “GGM” is GGM optimisation, and “M/T” stand for malicious or trusted setup.

	VPK Gen	VPK Verify	Prove	Verify	Proof size	VPK size
$DV_{\text{Prot}}$ M	4754	12310	162	66	5.52 KB	741 KB
$DV_{\text{Prot}}$ T	836	–	130	56	5.14 KB	159 KB
$DV_{\text{Prot}}$ M GGM	4754	12310	84	32	2.32 KB	741 KB
$DV_{\text{Prot}}$ T GGM	836	–	69	28	2.19 KB	159 KB
$DV_{\text{RangeProt}}$ M	13827	25900	1880	1120	34.32 KB	842 KB
$DV_{\text{RangeProt}}$ T	9106	–	1330	782	31.78 KB	188 KB
$DV_{\text{RangeProt}}$ M GGM	13827	25900	689	153	11.05 KB	842 KB
$DV_{\text{RangeProt}}$ T GGM	9106	–	490	111	10.41 KB	188 KB

**Theorem 5.** *Let AHE be an IND-CPA secure Encryption Scheme with message space  $|\mathcal{M}| > 2^{6\lambda+2\log\lambda+4}N_{\text{cm}}R$ . Then  $DV_{\text{RangeProt}}$  is a designated verifier argument of knowledge for the relation  $\mathcal{R}_{\text{HomRange}}$  that is: correct,  $Q$ -reusable knowledge-sound under the Factoring assumption for  $N_{\text{cm}}$  and IND-CPA security of AHE and statistically honest-verifier zero-knowledge under malicious VPK.*

Due to space limitations the proof is deferred to the full version.

$DV_{\text{RangeProt}}$  can be optimised in the generic group model similarly to how it is done in Sect. 4.6. In this case we can omit the final interaction between prover and verifier in Fig. 3 that proves knowledge of the plaintext inside  $S_i$ .

## 6 Evaluation and Performance

We implemented<sup>14</sup> and benchmarked our protocols, primarily focusing on evaluating and comparing  $DV_{\text{Prot}}$  and  $DV_{\text{RangeProt}}$  (Table 1), proving knowledge of the ciphertext message, and its range correspondingly. As a baseline we also implemented several flavours of the basic  $\Sigma$ -protocol (Table 2). For simplicity here we only present non-interactive (Fiat-Shamir transformed) variants.

The evaluation indicates that our protocols is a strictly better choice for certain types of applications (e.g. ID-MPC such as RSA ceremonies), as they exhibit better verification time and communication size. For *generic* applications, our protocols are comparable to other solutions, providing different performance trade-offs.

*Setup and Instantiation Details.* We ran our benchmarks on the Intel i5-8500 @ 3.00 GHz processor. For illustrative purposes the protocol code runs in the single-core mode only, and no specifically tailored low-level optimisations are implemented. All the evaluations are presented for  $\lambda = 128$ , and  $\log N = 2048$ ;

<sup>14</sup> The implementation is available publicly on Github: <https://github.com/volhovm/rsa-zkps-impl>.

**Table 2.** Performance for the baseline algorithms. Timings are in milliseconds.  $\text{Sigma}_{\text{Prot}}$  is evaluated with different  $p_{\text{max}}$ /number of repetition parameters. Note that  $\text{SigmaRange}_{\text{Prot}}$  has range slack while  $\text{DVRange}_{\text{Prot}}$  is tight.

	Prove	Pre-Verify	Verify	Proof size
$\text{Sigma}_{\text{Prot}}$ Paillier, $\lambda = 128$ reps	342	0	1161	134.00 KB
$\text{Sigma}_{\text{Prot}}$ Paillier, 8 reps	21	4	73	8.38 KB
$\text{Sigma}_{\text{Prot}}$ Paillier, 7 reps	19	36	64	7.33 KB
$\text{Sigma}_{\text{Prot}}$ Paillier, 6 reps	16	339	55	6.28 KB
$\text{Sigma}_{\text{Prot}}$ Paillier, 5 reps	14	6535	46	5.23 KB
$\text{SigmaRange}_{\text{Prot}}$ Paillier (with slack)	345	0	1157	108.00 KB

for the range proof we take  $R = 2^{256}$ ; the maximum query number of VPK reuses is set to  $Q = 128$ . For Fiat-Shamir transformation we instantiate the random oracle with the Blake2b [4] hash function.

For  $\text{DV}_{\text{Prot}}$  and  $\text{DVRange}_{\text{Prot}}$  we use Paillier-ElGamal encryption as the target homomorphism (which is additively homomorphic in both message and randomness), and standard Paillier as the AHE scheme on the verifier’s side. For each of our two protocols we evaluate four cases, depending on whether we use the GGM optimisation or not, and whether we consider malicious VPK or a trusted one (for the ID-MPC case). In the latter case we do not consider VPK verification time.

For the baseline  $\text{Sigma}_{\text{Prot}}$  and  $\text{SigmaRange}_{\text{Prot}}$  we use standard Paillier. We evaluate  $\text{Sigma}_{\text{Prot}}$  with naive  $\lambda = 128$  reps, and also with varying  $\log p_{\text{max}} \in \{16, 19, 22, 26\}$ . The range proof  $\text{SigmaRange}_{\text{Prot}}$  cannot use the  $p_{\text{max}}$  optimisation. Note, importantly, that  $\text{SigmaRange}_{\text{Prot}}$  has multiplicative range slack  $2^{\lambda+1}$ , while our  $\text{DVRange}_{\text{Prot}}$  is tight; this means comparing them directly is not even possible for all applications.

*Performance Overview.* Below we will mostly consider the GGM optimised variants of our protocols that assumes trusted setup, since it gives us best performance, and fits ID-MPC case well. The main advantage of our  $\text{DV}_{\text{Prot}}$  and  $\text{DVRange}_{\text{Prot}}$  is that they are single-shot, requiring no repetitions. It affects the two protocols non-proportionally, benefiting  $\text{DVRange}_{\text{Prot}}$  more, since the baseline  $\text{SigmaRange}_{\text{Prot}}$  cannot avoid  $\lambda$  repetitions. Our verification time is strictly less than the baseline: 1.5–2 $\times$  for  $\text{DV}_{\text{Prot}}$ , and 10 $\times$  for  $\text{DVRange}_{\text{Prot}}$ . Communication is more efficient too, since our proofs are strictly smaller. Even with our VPK being comparably heavy, its size together with  $Q = 128$  proofs gives us 1.5–2 $\times$  improvement for  $\text{DV}_{\text{Prot}}$  and 6–9 $\times$  improvement for  $\text{DVRange}_{\text{Prot}}$ . Our proving time is slightly higher for  $\text{DVRange}_{\text{Prot}}$ , and about 2 $\times$  higher with  $\text{DV}_{\text{Prot}}$ .

**Acknowledgements.** The first author received funding from projects from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), from the Spanish Government under project PRODIGY (TED2021-132464B-I00), and from the Madrid Regional Government under project BLOQUES (S2018/TCS-4339). The last two projects are co-funded by European Union EIE, and NextGenerationEU/PRTR funds. The last author was partially funded by Input Output ([iohk.io](http://iohk.io)) through their funding of the Edinburgh Blockchain Technology Lab.

## References

1. Attema, T., Cramer, R.: Compressed  $\Sigma$ -protocol theory and practical application to plug & play secure algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III, vol. 12172. LNCS, pp. 513–543. Springer, Heidelberg (2020). [https://doi.org/10.1007/978-3-030-56877-1\\_18](https://doi.org/10.1007/978-3-030-56877-1_18)
2. Attema, T., Cramer, R., Kohl, L.: A compressed  $\Sigma$ -protocol theory for lattices. In: Malkin, T., Peikert, C. (eds.) Annual International Cryptology Conference, CRYPTO 2021. LNCS, vol. 12826, pp. 549–579. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84245-1\\_19](https://doi.org/10.1007/978-3-030-84245-1_19)
3. Auerbach, B., Poettering, B.: Hashing solutions instead of generating problems: on the interactive certification of RSA moduli. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 403–430. Springer, Heidelberg (2018). [https://doi.org/10.1007/978-3-319-76581-5\\_14](https://doi.org/10.1007/978-3-319-76581-5_14)
4. Aumasson, J.-P., Neves, S., Wilcox-O’Hearn, Z., Winnerlein, C.: BLAKE2: simpler, smaller, fast as MD5. In: Jacobson Jr., M.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 119–135. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38980-1\\_8](https://doi.org/10.1007/978-3-642-38980-1_8)
5. Bangerter, E.: Efficient zero knowledge proofs of knowledge for homomorphisms. Ph.D. thesis. Citeseer (2005)
6. Bangerter, E., Camenisch, J., Krenn, S.: Efficiency limitations for  $\Sigma$ -protocols for group homomorphisms. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 553–571. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-11799-2\\_33](https://doi.org/10.1007/978-3-642-11799-2_33)
7. Bangerter, E., Camenisch, J., Maurer, U.: Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 154–171. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-30580-4\\_11](https://doi.org/10.1007/978-3-540-30580-4_11)
8. Bangerter, E., Krenn, S., Sadeghi, A.-R., Schneider, T., Tsay, J.-K.: On the design and implementation of efficient zero-knowledge proofs of knowledge. In: Software Performance Enhancements Encryption Decryption Cryptographic Compilers-SPEED-CC, vol. 9, pp. 12–13 (2009)
9. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997). [https://doi.org/10.1007/3-540-69053-0\\_33](https://doi.org/10.1007/3-540-69053-0_33)
10. Benhamouda, F., Ferradi, H., Géraud, R., Naccache, D.: Non-interactive provably secure attestations for arbitrary RSA prime generation algorithms. In: Foley, S.N., Gollmann, D., Sneekenes, E. (eds.) ESORICS 2017, Part I. LNCS, vol. 10492, pp. 206–223. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-66402-6\\_13](https://doi.org/10.1007/978-3-319-66402-6_13)

11. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC, pp. 103–112. ACM Press, May 1988. <https://doi.org/10.1145/62212.62222>
12. Böhl, F., Hofheinz, D., Jäger, T., Koch, J., Seo, J.H., Striecks, C.: Practical signatures from standard assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 461–485. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38348-9\\_28](https://doi.org/10.1007/978-3-642-38348-9_28)
13. 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](https://doi.org/10.1007/978-3-030-26948-7_20)
14. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45539-6\\_31](https://doi.org/10.1007/3-540-45539-6_31)
15. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-40061-5\\_3](https://doi.org/10.1007/978-3-540-40061-5_3)
16. Buchmann, J., Hamdy, S.: A survey on IQ cryptography (2001). <http://tubiblio.ulb.tu-darmstadt.de/100933/>
17. Bünz, D., 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>
18. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized Schnorr proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-01001-9\\_25](https://doi.org/10.1007/978-3-642-01001-9_25)
19. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-48910-X\\_8](https://doi.org/10.1007/3-540-48910-X_8)
20. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 413–430. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-48405-1\\_27](https://doi.org/10.1007/3-540-48405-1_27)
21. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020, pp. 1769–1787. ACM Press, November 2020. <https://doi.org/10.1145/3372297.3423367>
22. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-16715-2\\_26](https://doi.org/10.1007/978-3-319-16715-2_26)
23. Catalano, D., Pointcheval, D., Pornin, T.: IPAKE: isomorphisms for password-based authenticated key exchange. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 477–493. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_29](https://doi.org/10.1007/978-3-540-28628-8_29)
24. Chaidos, P., Couteau, G.: Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 193–221. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_7](https://doi.org/10.1007/978-3-319-78372-7_7)
25. Chan, A., Frankel, Y., Tsiounis, Y.: Easy come—easy go divisible cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054154>

26. Couteau, G., Kloof, M., Lin, H., Reichle, M.: Efficient range proofs with transparent setup from bounded integer commitments. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 247–277. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77883-5\\_9](https://doi.org/10.1007/978-3-030-77883-5_9)
27. Couteau, G., Peters, T., Pointcheval, D.: Removing the strong RSA assumption from arguments over the integers. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 321–350. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56614-6\\_11](https://doi.org/10.1007/978-3-319-56614-6_11)
28. Cramer, R.: Modular design of secure yet practical cryptographic protocols. Ph.D. thesis, CWI and University of Amsterdam (1996)
29. Cramer, R., Damgård, I.: On the amortized complexity of zero-knowledge protocols. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 177–191. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03356-8\\_11](https://doi.org/10.1007/978-3-642-03356-8_11)
30. Cramer, R., Damgård, I., Keller, M.: On the amortized complexity of zero-knowledge protocols. *J. Cryptol.* **27**(2), 284–316 (2013). <https://doi.org/10.1007/s00145-013-9145-x>
31. 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](https://doi.org/10.1007/3-540-46035-7_4)
32. Damgård, I.: On  $\Sigma$ -Protocols. Lecture Notes, University of Aarhus, Department for Computer Science, p. 84 (2002). Accessed: 16 Feb 2022
33. Damgård, I., Fazio, N., Nicolosi, A.: Non-interactive zero-knowledge from homomorphic encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 41–59. Springer, Heidelberg (2006). [https://doi.org/10.1007/11681878\\_3](https://doi.org/10.1007/11681878_3)
34. 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](https://doi.org/10.1007/3-540-36178-2_8)
35. Damgård, I., Jurik, M.: A length-flexible threshold cryptosystem with applications. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 350–364. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-45067-X\\_30](https://doi.org/10.1007/3-540-45067-X_30)
36. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44586-2\\_9](https://doi.org/10.1007/3-540-44586-2_9)
37. 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](https://doi.org/10.1007/3-540-46035-7_17)
38. Dobson, S., Galbraith, S.D., Smith, B.: Trustless groups of unknown order with hyperelliptic curves. *Cryptology ePrint Archive*, Report 2020/196 (2020). <https://eprint.iacr.org/2020/196>
39. Fujisaki, E., Okamoto, T.: A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 32–46. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054115>
40. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052225>



41. Gennaro, R., Micciancio, D., Rabin, T.: An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In: Gong, L., Reiter, M.K. (eds.) ACM CCS 1998, pp. 67–72. ACM Press, November 1998. <https://doi.org/10.1145/288090.288108>
42. Goldberg, S., Reyzin, L., Sagga, O., Baldimtsi, F.: Efficient noninteractive certification of RSA moduli and beyond. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 700–727. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34618-8\\_24](https://doi.org/10.1007/978-3-030-34618-8_24)
43. Goldwasser, S., Kharchenko, D.: Proof of plaintext knowledge for the Ajtai-Dwork cryptosystem. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 529–555. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-30576-7\\_29](https://doi.org/10.1007/978-3-540-30576-7_29)
44. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005). [https://doi.org/10.1007/11496137\\_32](https://doi.org/10.1007/11496137_32)
45. Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T., Nicolosi, A.A.: Efficient RSA key generation and threshold Paillier in the two-party setting. *J. Cryptol.* **32**(2), 265–323 (2018). <https://doi.org/10.1007/s00145-017-9275-7>
46. Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 369–386. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44381-1\\_21](https://doi.org/10.1007/978-3-662-44381-1_21)
47. Kirchner, P., Fouque, P.-A.: Getting rid of linear algebra in number theory problems. *Cryptology ePrint Archive*, Report 2020/1619 (2020). <https://ia.cr/2020/1619>
48. Kosba, A., Papamanthou, C., Shi, E.: xJsnark: a framework for efficient verifiable computation. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 944–961. IEEE (2018)
49. Kunz-Jacques, S., Martinet, G., Poupard, G., Stern, J.: Cryptanalysis of an efficient proof of knowledge of discrete logarithm. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 27–43. Springer, Heidelberg (2006). [https://doi.org/10.1007/11745853\\_3](https://doi.org/10.1007/11745853_3)
50. Lee, J.: The security of groups of unknown order based on Jacobians of hyperelliptic curves. *Cryptology ePrint Archive*, Report 2020/289 (2020). <https://eprint.iacr.org/2020/289>
51. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-40061-5\\_26](https://doi.org/10.1007/978-3-540-40061-5_26)
52. Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
53. Ozdemir, A., Wahby, R., Whitehat, B., Boneh, D.: Scaling verifiable computation using efficient set accumulators. In: 29th USENIX Security Symposium (USENIX Security 2020), pp. 2075–2092 (2020)
54. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-48910-X\\_16](https://doi.org/10.1007/3-540-48910-X_16)
55. Pass, R., Shelat, A., Vaikuntanathan, V.: Construction of a non-malleable encryption scheme from any semantically secure one. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 271–289. Springer, Heidelberg (2006). [https://doi.org/10.1007/11818175\\_16](https://doi.org/10.1007/11818175_16)

56. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 187–196. ACM Press, May 2008. <https://doi.org/10.1145/1374376.1374406>
57. Rabin, M.O., Shallit, J.O.: Randomized algorithms in number theory. *Commun. Pure Appl. Math.* **39**(S1), S239–S256 (1986)
58. 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](https://doi.org/10.1007/0-387-34805-0_22)
59. 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](https://doi.org/10.1007/3-540-69053-0_18)
60. Terelius, B., Wikström, D.: Efficiency limitations of  $\Sigma$ -protocols for group homomorphisms revisited. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 461–476. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32928-9\\_26](https://doi.org/10.1007/978-3-642-32928-9_26)
61. van de Graaf, J., Peralta, R.: A simple and secure way to show the validity of your public key. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 128–134. Springer, Heidelberg (1988). [https://doi.org/10.1007/3-540-48184-2\\_9](https://doi.org/10.1007/3-540-48184-2_9)
62. Yuen, T.H., Huang, Q., Mu, Y., Susilo, W., Wong, D.S., Yang, G.: Efficient non-interactive range proof. In: Ngo, H.Q. (ed.) COCOON 2009. LNCS, vol. 5609, pp. 138–147. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-02882-3\\_15](https://doi.org/10.1007/978-3-642-02882-3_15)



# Dew: A Transparent Constant-Sized Polynomial Commitment Scheme

Arasu Arun<sup>1</sup>, Chaya Ganesh<sup>2</sup>, Satya Lokam<sup>3</sup>, Tushar Mopuri<sup>2</sup>(✉),  
and Sriram Sridhar<sup>4</sup>

<sup>1</sup> New York University, New York, USA  
`arasu@nyu.edu`

<sup>2</sup> Indian Institute of Science, Bengaluru, India  
`{chaya,tusharmopuri}@iisc.ac.in`

<sup>3</sup> Microsoft Research India, Bengaluru, India  
`satya@microsoft.com`

<sup>4</sup> University of California, Berkeley, USA  
`srirams@berkeley.edu`

**Abstract.** We construct a polynomial commitment scheme with constant (i.e., independent of the degree) sized evaluation proofs and logarithmic (in the degree) verification time in the transparent setting. To the best of our knowledge, this is the first result achieving this combination of properties.

We build our scheme from an inner product commitment scheme with constant-sized proofs but with linear verification time. To improve the verification time to logarithmic for polynomial commitments, we prove a new extremal combinatorial bound. Our constructions rely on groups of unknown order instantiated by class groups. We prove security of our constructions in the Generic Group Model.

Compiling known information-theoretic proof systems using our polynomial commitment scheme yields transparent and constant-sized zkSNARKs (Zero-knowledge Succinct Non-interactive ARGuments of Knowledge) with logarithmic verification.

## 1 Introduction

A Polynomial Commitment Scheme (PCS) [18] allows a prover to commit to a polynomial  $P$  of degree  $d$  so that, later, a verifier can query for  $P(x)$  at an argument  $x$  of its choice and the prover can, together with its response, furnish an evaluation proof that its response is indeed consistent with its commitment. The commitment and the evaluation proof are required to be *succinct*, that is, of size independent of, or logarithmic, in  $d$ .

Polynomial commitments have applications in verifiable secret sharing [15], anonymous credentials [10], and zero-knowledge sets [22], among others. But by

---

S. Sridhar, T. Mopuri and A. Arun—Work partially done while at Microsoft Research India.

far, their most dominant application is to constructions of zkSNARKs (zero-knowledge Succinct Non-interactive ARGuments of Knowledge) for all of NP. Indeed, improvements to PCS imply improvements to SNARKs when combined with established modular approaches to SNARKs. On the other hand, a SNARK for all of NP in particular implies a succinct PCS by instantiating the SNARK for the NP-relation “ $y = P(x)$  and the commitment  $C$  opens to  $P$ ”, where  $C$  is a succinct commitment to  $P$ . While this incidental corollary of a SNARK implies succinct PCS, we desire a “direct” construction of PCS without writing polynomial evaluation as a generic NP relation, since most SNARK constructions themselves use PCS as a crucial ingredient. PCS is therefore a core cryptographic construct and there is a strong motivation to construct one with the best possible parameters.

*Transparent Setup.* Non-interactive proof systems are typically in the Common Reference String (CRS) model where a CRS is generated during a setup phase which needs to be *trusted* if the CRS uses secret randomness. Constructions that do not involve a trusted setup phase and the verifier randomness consists of only *public coins* are called *transparent*. A recent line of work [11, 12, 17, 23] to construct SNARKs follows a modular approach: first, an information-theoretic component is constructed; then this is compiled into an argument using cryptographic tools, typically a PCS. Finally, this is made non-interactive to obtain a SNARK in the random oracle model (ROM). The resulting SNARK inherits the trusted setup assumption or the transparency property from the cryptographic tools used in the compilation process. Any resulting SNARK from compiling an information-theoretic protocol inherits the complexity of the PCS, that is, the proof size depends on the commitment size and evaluation proof size of the PCS. Unfortunately, all existing succinct PCS schemes either require trusted-setup assumptions [18], or are when they are transparent, only achieve logarithmic proof size [3, 9, 20]<sup>1</sup>. We address this challenge in this work.

## 1.1 Our Contributions

We present the first PCS with constant size commitment, constant size evaluation proof<sup>2</sup> and logarithmic verification in the transparent setting. Our starting point is a construction of a transparent Inner Product Commitment (IPC) scheme (which is a more general object than PCS) that allows a prover to open a committed vector to inner products with a verifier’s query vectors. Our IPC is succinct – that is, the size of the commitment and the proof of a correct opening are independent of the length of the vector and linear (in the length of the vector) time verification. Building on this IPC, we construct a succinct PCS,

<sup>1</sup> A flaw in the proof of security of the DARK scheme [9] was discovered by Block et al [3], who propose a different PCS with logarithmic proof size. We also note that a revised version of DARK [8] also proposes a fix by showing that the DARK PCS satisfies a property called almost-special-soundness which suffices for extraction.

<sup>2</sup> Constant is  $O_\kappa(1)$ . That is, independent of the size of the input, and polynomial only in the security parameter.

resulting in a *transparent constant-sized PCS* but, importantly, with *logarithmic time verification*. Our PCS is the *first* construction to achieve the above combination of properties to the best of our knowledge.

From a technical point of view, our contributions are summarized below.

**Inner Product Commitment (IPC) and Polynomial Commitment Scheme (PCS).** We construct a constant size transparent IPC scheme in Sect. 3. In Sect. 4, we present our transparent PCS construction that achieves constant sized proofs, constant sized public parameters, and verification in  $O(\log n)$  field operations and a constant number of group operations for polynomials of degree  $n$ . Both the above constructions are in the GGM. We also show hiding and zero knowledge variants of our constructions. Using the now standard compilation process from information-theoretic proofs in idealized models to zkSNARKs via PCS [9, 12], we obtain a *transparent constant-sized zkSNARK with constant-sized public parameters* (Sect. 5). The resulting zkSNARKs achieve  $O_\kappa(1)$  communication and  $O(\log n)$  verification<sup>3</sup>, where  $n$  is the complexity of the NP relation (e.g., number of constraints of a Rank 1 Constraint System, or the number of gates in an arithmetic circuit). The only other transparent zkSNARKs with constant-sized proofs and public parameters are obtained by compiling constant-query PCPs using transparent vector commitment schemes with constant-sized opening proofs and public parameters. The VCs of [5, 19] are such candidates.

**A New Combinatorial Lemma.** As noted above, we improve the verification time from linear (in length of vector) in our IPC to logarithmic (in degree of polynomial) in our PCS. We achieve this efficiency improvement using Kronecker products (details in Sect. 1.3.2 and Sect. 4.1); but their naive application breaks soundness. We recover soundness by solving a problem in extremal combinatorics. A special case of our problem asks: how many points can we choose in the discrete cube  $[n]^d$  such that that set of points does not contain the corners of a  $d$ -dimensional hyper-rectangle (box)? When  $d = 2$ , this is the Zarankiewicz problem [4] in extremal graph theory for which an asymptotically tight bound of  $\sim n^{3/2}$  is known. For higher  $d$ , the “Box Theorem” due to Rosenfeld [24] proves the bound  $\sim n^{d-2^{-d+1}}$ . We can use these bounds in the soundness proof of our PCS to obtain  $n^\epsilon$  verification time for any constant  $0 < \epsilon < 1$ . While this improves on linear, our goal is to obtain *logarithmic* verification.

We achieve logarithmic verification by generalizing boxes in the extremal problem to “d-cancellation structures.” With these d-cancellation structures, we can continue to exploit certain cancellation properties required for soundness (details in Sect. 4.2) similar to those for boxes but also, more importantly, succeed in proving much better bounds on the number of points in  $[n]^d$  that do not contain a d-cancellation structure. Our bounds are  $\sim dn^{d-1}$  and it is crucial for our soundness that this is a negligible fraction of the whole space (of size  $n^d$ ); in contrast, as  $d \rightarrow \infty$ , the box theorem above gives a bound that approaches  $n^d$  – essentially filling the whole space.

---

<sup>3</sup> In the preprocessing setting.

To the best of our knowledge, our result is the first application of an extremal combinatorics theorem in the construction of PCS and SNARKs and we believe this to be of independent interest. We note that extremal combinatorics results like this have found applications in complexity theory and theoretical computer science in general.

**Recovering the DARK [9] Result.** We show that our PCS can be adapted to obtain logarithmic proof size and verification by employing the recursive evaluation protocol from DARK on our new commitment scheme. This recovers the flawed Lemmas 8, 9 from DARK thus recovering a transparent PCS with logarithmic proof size and logarithmic verification, *but* at the expense of an increased quadratic prover time. The DARK recovery does not require GGM; we achieve this result under the same assumptions made in DARK, i.e., the Adaptive Root and Strong RSA Assumptions. We note that [3] gives a construction that achieves similar results as DARK by modifying DARK’s evaluation protocol, and a subsequent revision of DARK [8] shows that the DARK PCS satisfies a property called almost-special-soundness. In contrast, our construction is a commitment scheme that is syntactically close to DARK, has a similar evaluation protocol and recovers the flawed lemmas. We present this in the full version [1].

## 1.2 Related Work

Functional commitments were introduced by [21] as a generalization of vector commitments, where a prover can commit to a vector, and later open the commitment at functions of the committed vector with a succinct proof that the answer is consistent with the committed vector. The work of [21] also showed a construction for functional commitments for linear functions. Lai and Malavolta [19] put forth the notion of Linear Map Commitments (LMC) that allow a prover to open a commitment to the output of a linear map. The constructions from [19, 21] achieve succinctness - constant commitment and proof size, but require trusted setup.

In a recent concurrent work, [13] presents transparent inner product commitment schemes with constant size openings and constant size public parameters. Their scheme is also in groups of unknown order, however, the techniques they use are completely different. Their result relies on proofs of cardinality of RSA accumulated sets, whereas we rely on integer encoding of vectors and combinatorial techniques to show extraction. Though a PCS was not their goal, a PCS resulting from the inner product commitment scheme of [13] in the natural way results in a linear time verifier. In contrast, we achieve logarithmic verification time for our PCS.

Polynomial commitment schemes were introduced in [18], and have since led to several variants being used in recent SNARKs. The KZG scheme [18] gives constant-sized commitments and proofs, but require a trusted setup. In the transparent setting, Wahby et al. [25] constructed a polynomial commitment scheme for multilinear polynomials that has commitment size and evaluation proof size  $O(\sqrt{d})$  for degree  $d$  polynomials. Zhang et al. [27] construct a polynomial com-

mitment from FRI (Fast Reed Solomon IOPP) that is transparent, has constant size commitments, but evaluation proofs have size  $O(\log^2 d)$ .

As mentioned earlier, Bünz et al [9] used a Diophantine Argument of Knowledge (DARK), and constructed a polynomial commitment scheme with proof size  $O(\log d)$  and  $O(\log d)$  verification time for polynomials of degree  $d$ . Block et al. [3] identified a gap in the proof of security of the DARK scheme and propose a modification that sidesteps the gap in extraction, resulting in a PCS of polylogarithmic proof size and verification time.

### 1.3 Technical Overview

The intuitive starting point of our commitment schemes is a natural mapping from vectors to group elements *via* integers. Specifically, for a vector<sup>4</sup>  $\mathbf{c}$ , define  $\text{int}_\alpha(\mathbf{c}) := \langle \mathbf{c}, \boldsymbol{\alpha} \rangle := \sum_0^{l-1} c_i \alpha^i$ , where  $\boldsymbol{\alpha} := (1, \alpha, \alpha^2, \dots, \alpha^{l-1})$  and  $\alpha$  is sufficiently large. Let us also define  $C := g^{\text{int}_\alpha(\mathbf{c})}$  for a given group element  $g \in \mathbb{G}$ . When the group  $\mathbb{G}$  is a *group of unknown order* and  $g \in \mathbb{G}$  is random (but chosen during set up), we can show that a prover can prove knowledge of a unique positive exponent of  $C$  with base  $g$  and that the  $\alpha$ -base representation  $\mathbf{c}$  of that exponent must be a valid opening of  $C$ . This follows from a Proof of Knowledge of a *Positive Exponent* (PoKPE) protocol that builds on Wesolowski’s Proof of Exponent (PoE) protocol [26] (details in Sect. 2.4).

We now wish to use the commitment  $C$  made by a Prover to vector  $c$  in a protocol for inner product  $\langle \mathbf{c}, \mathbf{q} \rangle$ , where  $\mathbf{q}$  is the *query vector* from the Verifier. To that end, let us consider the integer product

$$\text{int}_\alpha(\mathbf{c}) \cdot \text{int}_\alpha(\text{reverse}(\mathbf{q})) = \left( \sum_{i=0}^{l-1} c_i \alpha^i \right) \cdot \left( \sum_{i=0}^{l-1} q_{l-i} \alpha^i \right) = L + \alpha^l \langle \mathbf{c}, \mathbf{q} \rangle + H \quad (1)$$

where  $L$  and  $H$  are polynomials collecting powers of  $\alpha$  of degree less than  $l$  and more than  $l$  respectively. Raising  $g$  to both sides of (1), we obtain

$$C^{\text{int}_\alpha(\text{reverse}(\mathbf{q}))} = (g^L) \cdot (g^{\alpha^l \langle \mathbf{c}, \mathbf{q} \rangle}) \cdot (g^H). \quad (2)$$

Note that the verifier can compute the l.h.s. of (2). Hence if the prover claims that the inner product  $\langle \mathbf{c}, \mathbf{q} \rangle$  evaluates to  $v$  and *also* sends  $g^L$  and  $g^H$  to the verifier (and convinces the verifier of values  $L$  and  $H$  using a PoKPE protocol), then the verifier can check consistency of the prover’s claim using (2) (with  $v$  in place of  $\langle \mathbf{c}, \mathbf{q} \rangle$ ). While this intuition suffices for a completeness proof, it is by no means sufficient for a soundness proof. Our main contribution, outlined below, is to show that a check somewhat analogous to (2) with some additional machinery *suffices* for a verifier to catch a cheating prover with high probability. This intuition is essentially the basis for our inner product evaluation protocol **IPP** in Fig. 3 and the “additional machinery” appears as the **TEST** protocol in Fig. 2.

We remark that the above intuition is reminiscent of approaches in [9, 19]. However, our approach below differs significantly from theirs to achieve *constant*

---

<sup>4</sup> Our vectors are over  $\mathbb{Z}_p$  and we map elements of  $\mathbb{Z}_p$  to integers  $\{0, \dots, p - 1\}$ .

sized proofs (unlike in [9] that uses recursion to obtain logarithmic size) and transparent setting where  $\alpha$  is not secret (unlike the trusted setup in [19]).

**1.3.1 A TEST Protocol and Extracting Structure from Overflows**

A cheating prover could use a committed vector (derived by computing the  $\alpha$ -base representation of exponent of  $g$  in  $C$  using the PoKPE extractor) with coordinate values that could cause “overflow” in the coefficients on the r.h.s. of (1). In that case, we can no longer guarantee the correctness of the inner product as the middle coefficient. We now describe ideas that help us overcome this challenge.

To control the issues caused by overflow, we intersperse 0’s between coordinates of  $\mathbf{c}$ : we double the length of the vector to  $2l$  and place the vector  $\mathbf{c}$  to be committed in even positions  $(0, 2, \dots, 2l - 2)$  and 0’s in odd positions. More generally, let  $\mathbf{d}$  denote the subvector in the odd positions and let  $\mathbf{c}||\mathbf{d}$  denote the combined vector of length  $2l$ . Note first that completeness continues to hold with this change since an honest prover commits to  $\mathbf{c}||\mathbf{0}$ , with  $\mathbf{c} \in \mathbb{Z}_p^l$  and satisfy analogs of (1) and (2) for length- $2l$  vectors with 0’s in odd positions, or equivalently, with  $\alpha^2$  replacing  $\alpha$ . Second, and this is our next crucial step, note that the verifier can run a **TEST** protocol (cf. Fig 2) that queries for the inner product  $\langle (\mathbf{c}||\mathbf{d}), (\mathbf{0}||\mathbf{z}) \rangle$ , where  $\mathbf{z} \in \mathbb{Z}_p^l$  is a uniformly chosen *random* vector and the verifier can check if the middle coefficient of (generalization to  $2l$ -length vectors of) (2) is zero. *Assuming no overflows*, the middle coefficient would be  $\langle \mathbf{d}, \mathbf{z} \rangle$ . In this case, by the Schwartz-Zippel lemma, a cheating prover choosing nonzero  $\mathbf{d}$  would be caught with high probability. However, a cheating prover could choose nonzero  $\mathbf{d}$  and still pass the test for  $\mathbf{z}$  by causing overflows from  $L$  in (1) (due to large products between  $\mathbf{c}$  and  $\mathbf{z}$  coordinates) to “cancel out” the non-zero value of  $\langle \mathbf{d}, \mathbf{z} \rangle$ . More precisely, it can be shown using (a generalization of) (1) and the PoKPE protocol relating (2) to (1) that if a prover can succeed in the **TEST** protocol with non-negligible probability, then

$$\langle \mathbf{d}, \mathbf{z} \rangle \bmod \alpha + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u = 0 \bmod \alpha, \tag{3}$$

for some  $u \in \{0, 1\}$  must be satisfied (3) with non-negligible probability over a uniformly random  $\mathbf{z} \in \mathbb{Z}_p^l$ .

Call a test vector  $\mathbf{z} \in \mathbb{Z}_p^l$  a *success point* for the prover if (3) is satisfied when verifier chooses  $\mathbf{z}$  and prover’s commitment  $C$  extracts – via a PoKPE protocol – to  $\mathbf{c}||\mathbf{d}$ . Now, if a prover has two success points  $\mathbf{z}$  and  $\mathbf{z}'$  on a “line”, i.e.,  $\mathbf{z}$  and  $\mathbf{z}'$  agree on all coordinates except  $j$ -th, then  $\langle \mathbf{d}, \mathbf{z} \rangle - \langle \mathbf{d}, \mathbf{z}' \rangle = d_j(z_j - z'_j)$  because of cancellations in coordinates  $\neq j$ . Thus subtracting (3) for  $\mathbf{z}'$  from that for  $\mathbf{z}$ , we obtain

$$d_j \cdot (z_j - z'_j) = \left( \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}' \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u' - u \right) \bmod \alpha. \tag{4}$$

Using bounds on coordinates of  $\mathbf{c}$  and  $\mathbf{z}$ , we conclude that  $d_j$  is  $\theta_{1j}\alpha + \theta_{2j}$ , where  $\theta_1$  and  $\theta_2$  are rationals with small denominators (a more detailed statement of



this structure appears in Theorem 34). An easy combinatorial argument shows that if the prover succeeds with non-negligible probability, e.g., at least  $1/p$  ( $p$  is  $\exp(\kappa)$ ), then in *every* direction  $j$ , there must be a line in the  $j$ -th dimension with two success points  $\mathbf{z}$  and  $\mathbf{z}'$  on it. Hence, if the prover is accepted in the **TEST** protocol (Fig. 2) with non-negligible probability, every coordinate of  $\mathbf{d}$  can be expressed as  $d_j = \theta_{1j}\alpha + \theta_{2j}$  with  $\theta$ 's as above – *this is the structure we extract on  $\mathbf{d}$  that we use to prove soundness*. This Structure Theorem 34 is a crucial technical ingredient of our results.

Armed with the structure theorem, we prove (Theorem 35) that if the inner product evaluation protocol **IPP** (Fig. 3) for  $\langle\langle \mathbf{c} \parallel \mathbf{d} \rangle\rangle, (\mathbf{q} \parallel \mathbf{0})$  succeeds in satisfying (generalizations) of (1) and (2), with query vector  $\mathbf{q} \parallel \mathbf{0}$ , then we can extract a vector  $\tilde{\mathbf{c}}$  that, while fractional over the integers, has invertible denominators modulo  $p$ . Using this  $\tilde{\mathbf{c}}$  as the “opening hint” (cf. **Open()** in §3.1), we can then extract a unique  $\mathbf{c}$  that is consistent with the claimed inner product.

### 1.3.2 Logarithmic Verification for Polynomial Commitments

Our IPC scheme above immediately yields a Polynomial Commitment Scheme (PCS), noting that, for a polynomial  $f$  given by its vector of coefficients  $\mathbf{f} = (f_0, \dots, f_{l-1})$ ,  $f(x) = \langle \mathbf{f}, \mathbf{x} \rangle$ , where  $\mathbf{x} = (1, x, \dots, x^{l-1})$ . However, the verification complexity of the resulting PCS is much worse than what we want to achieve. Linear verification seems inherent for inner products (since the query vector  $\mathbf{q}$  can be arbitrary and the verifier needs to at least read the statement, verifier’s computation of  $\text{int}_\alpha(\text{reverse}(\mathbf{q}))$  itself will take linear time). But, in a PCS, we can hope to achieve logarithmic verification time since the query vector  $\mathbf{x}$  is parameterized by single variable  $x$ . In particular, we can compute  $\text{int}_\alpha(\text{reverse}(\mathbf{x}))$  in only logarithmic time (cf. (5) below and (11)). This makes the verifier in **IPP** protocol (specialized to a PCS) logarithmic. However, we still have the bottleneck for verifier computation in the **TEST** protocol. Note that while the query vector  $\mathbf{x}$  is parameterized by a single variable  $x$ , the *test vector*  $\mathbf{z}$  is not and hence computing  $\text{int}_\alpha(\text{reverse}(\mathbf{z}))$  in checking (2) in **TEST** protocol still seems to require linear verifier time.

To reduce verifier’s computation in **TEST** protocol, we use the idea of **Kronecker products**<sup>5</sup>: instead of choosing  $\mathbf{z}$  uniformly at random in  $\mathbb{Z}_p^l$ , we choose  $\log l$  vectors  $\mathbf{z}_0, \dots, \mathbf{z}_{\log l-1}$  uniformly at random from  $\mathbb{Z}_p^2$  and define  $\mathbf{z} = \mathbf{z}_0 \otimes \dots \otimes \mathbf{z}_{\log l-1}$ . To illustrate how this helps, consider the following computation needed on the right hand side of (2), where  $\mathbf{z}$  is as above and  $i = (i_0, \dots, i_{\log l-1})$  the binary expansion of index  $i \in [l]$ .

$$\text{int}_\alpha(\text{reverse}(\mathbf{z})) = \sum_{i=0}^{l-1} \alpha^{l-i} \prod_{j=0}^{\log l-1} z_{j,i_j} = \alpha^l \cdot \prod_{j=0}^{\log l-1} (z_{j,0} + z_{j,1}\alpha^{-2^j}), \quad (5)$$

---

<sup>5</sup> We note that [2] and [20] also use Kronecker products in proof systems albeit with different motivations.

and note that the last product can be computed in logarithmic time. The new test protocol with Kronecker product test vectors is called **logTEST** (identical to **TEST** except with the query vector replaced as above).

While this helps improve verifier efficiency of **TEST**, it *breaks soundness!* The extractability proof of **TEST** in IPC relies on uniform randomness of the test vector  $\mathbf{z} \in \mathbb{Z}_p^l$ . So, we must now *improve the extractability proof to work with exponentially smaller randomness* in the  $\log l$  vectors  $\mathbf{z}_j$  of length 2. Specifically, it is crucial to recover an analog of the structure for the  $\mathbf{d}$  vector as outlined in Sect. 1.3.1 but now from this vastly reduced space of verifier’s randomness in **logTEST**. We outline how to do this next.

### 1.3.3 An Extremal Combinatorial Bound

We recover soundness with Kronecker product **TEST** vectors by proving a *new result in extremal combinatorics*. Informally, this theorem (Theorem 45) gives a tight upper bound on the number of points in the hypercube  $[n]^d$  such that no subset of  $2^d$  points in that set form a configuration that we call a **d-cancellation structure**. A **d-cancellation structure** generalizes the set of corners of a  $d$ -dimensional *box* or a hyper-rectangle. For instance, a **2-cancellation structure** is a parallelogram generalizing a rectangle. In the case of a rectangle, this is the well-known Zarankiewicz problem [4] from extremal graph theory and has an asymptotically tight bound of  $n^{3/2}$  points (out of  $n^2$ ) that contain no four points as corners of a rectangle in  $[n]^2$ . Thus, our problem generalizes this in two ways: first, we consider high dimensions with growing  $d$  (but no more than  $\log n$ ) and second, we generalize a rectangle/box to **d-cancellation structure**. A recursive definition is given in Definition 44. For  $d > 2$  and in case of boxes, Rosenfeld [24] proved an upper bound of  $\sim n^{d-2^{-d+1}}$  on the maximum number of points that do not contain the corners of a box. This bound, however, is insufficient for us to get logarithmic verification since as  $d$  grows, it tends to  $n^d$  almost entirely filling the space. Our main contribution is to obtain a significantly smaller upper bound by generalizing boxes to **d-cancellation structure**: a tight upper bound of  $(n^d - (n-1)^d) \leq dn^{d-1}$ , which is a vanishingly small fraction of  $n^d$ . For example, when the forbidden configurations are generalized from rectangles to parallelograms for  $d = 2$ , the upper bound improves to  $\sim n$  from the  $\sim n^{3/2}$  stated above for the Zarankiewicz problem.

We now tie back this combinatorial argument to the goal of extractability. As the name implies, a **d-cancellation structure** induces cancellations. Recall from Sect. 1.3.1 that cancellations between two success points (test vectors  $\mathbf{z}$  and  $\mathbf{z}'$  where the prover succeeds by satisfying (3)) allow us to deduce structural conditions on coordinates of  $\mathbf{d}$  using (4). We now generalize that argument to Kronecker products as test vectors where a **d-cancellation structure** generalizes the role of a line, and cancellations between two points on a line generalize to recursive cancellations among  $2^d$  points in a **d-cancellation structure**. Finally, the simple combinatorial argument outlined in Sect. 1.3.1 for **TEST** on the existence of at least one line in each dimension with at least two success points is replaced by the existence of a **d-cancellation structure** for every index  $i = (i_0, \dots, i_{\log l-1})$

(corresponding to a  $\mathbf{d}$ -coordinate  $d_i$ , cf. (5)) in the Kronecker product space for **logTEST**.

Specifically, each accepting run of **logTEST** corresponds to a chosen/success point in  $[n]^d$  (this is our space of randomness, with  $n = p$  and  $d = \log l$ ). By suitable calibration of parameters, we can show that a prover that succeeds with a non-negligible probability gives rise to more than  $n^d - (n - 1)^d$  chosen points and then our combinatorial bound above implies the existence of a  $\mathbf{d}$ -cancellation structure  $B$ , each of whose ‘‘corners’’ (for simplicity, think of a  $\mathbf{d}$ -cancellation structure as a box) is a success point. Thus, we obtain  $2^d$  equations like (3) at the corners of  $B$ ;  $\langle \mathbf{d}, \mathbf{z} \rangle$  is a multilinear polynomial with coefficients  $d_i$  ( $i$ -th coordinate of  $\mathbf{d}$ , with bit representation of  $i = (i_0, \dots, i_{\log l-1})$ ) and variables  $z_{j,i_j}$  (cf. (5)) from the Kronecker product **TEST** vector  $\mathbf{z} = \otimes_{j=0}^{\log l-1} \mathbf{z}_j$ . The recursive structure of  $B$  allows recursively combining these equations by folding, i.e., subtracting equations like (3) along ‘‘edges’’ of  $B$  in the same direction. Each successive folding reduces the number of equations by half and eliminates one of the free variables  $z_{j,i_j}$  to obtain a multilinear version of (4). After  $\log l$  such folding steps, we obtain an equation generalizing (4) with one  $d_i$  on l.h.s, that yields the structure on coordinates of  $\mathbf{d}$  that we seek. This helps us recover an analog of the structure theorem (Theorem 34) for **logTEST** (Theorem 42).

## 2 Preliminaries

*Notation.* A finite field is denoted by  $\mathbb{F}$ . We denote by  $\kappa$  a security parameter. When we explicitly specify the random tape for a randomized algorithm  $A$ , then we write  $a \leftarrow A(\mathbf{pp}; \rho)$  to indicate that  $A$  outputs  $a$  given input  $\mathbf{pp}$  and random tape  $\rho$ . We consider interactive arguments for relations, where a prover  $P$  convinces the verifier that it knows a witness  $w$  such that for a public statement  $x$ ,  $(x, w) \in \mathcal{R}$ . For a pair of PPT interactive algorithms  $P, V$ , we denote by  $\langle P(w), V \rangle(x)$ , the output of  $V$  on its interaction with  $P$  where  $w$  is  $P$ 's private input and  $x$  is a common input.

*Fiat-Shamir transform.* In this work, we consider *public coin* interactive arguments where the verifier's messages are uniformly random strings. Public coin protocols can heuristically be made non-interactive by applying the Fiat-Shamir [16] transform (FS) in the Random Oracle Model (ROM).

### 2.1 Inner Product Commitments

We define Inner Product Commitments (IPC) which is an extension of functional commitments introduced in [21]. IPC allows a prover to prove that the committed vector  $\mathbf{f}$  satisfies  $\langle \mathbf{f}, \mathbf{q} \rangle = v$ , for some vector  $\mathbf{q}$  and  $v$ .

An Inner Product Commitment scheme over  $\mathbb{F}$  is a tuple  $\text{IPC} = (\mathbf{Setup}, \mathbf{Com}, \mathbf{Open}, \mathbf{Eval})$  where:

- $\mathbf{Setup}(1^\kappa, D) \rightarrow \text{pp}$ . On input security parameter  $\kappa$ , and an upper bound  $D$  on accepted vector lengths,  $\mathbf{Setup}$  generates public parameters  $\text{pp}$ .
- $\mathbf{Com}(\text{pp}, f_0, \dots, f_{l-1}, l) \rightarrow (C, \tilde{\mathbf{c}})$ . On input the public parameters  $\text{pp}$ , the length of the vector  $l \leq D$  and a vector of length  $l$ , given as  $f_0, \dots, f_{l-1} \in \mathbb{F}$ ,  $\mathbf{Com}$  outputs a commitment  $C$ , and additionally an opening hint  $\tilde{\mathbf{c}} \equiv (f_0 \dots, f_{l-1})$ .
- $\mathbf{Open}(\text{pp}, \mathbf{f}, l, C, \tilde{\mathbf{c}}) \rightarrow b$ . On input the public parameters  $\text{pp}$ , the opening hint  $\tilde{\mathbf{c}}$ , the length of the vector in the commitment  $l$  and the commitment  $C$ , the claimed committed vector  $\mathbf{f}$ ,  $\mathbf{Open}$  outputs a bit indicating accept or reject.
- $\mathbf{Eval}(\text{pp}, C, l, \mathbf{q}, v; \mathbf{f}) \rightarrow b$ . A public coin interactive protocol  $\langle P_{\mathbf{Eval}}(\mathbf{f}), V_{\mathbf{Eval}} \rangle(\text{pp}, C, l, \mathbf{q}, v)$  between a PPT prover and a PPT verifier. The parties have as common input public parameters  $\text{pp}$ , commitment  $C$ , the length of the vector in the commitment  $l$ , query vector  $\mathbf{q} \in \mathbb{F}^l$ , and claimed inner product  $v$ . The prover has, in addition, the vector committed to in  $C$ ,  $\mathbf{f}$ . At the end of the protocol, the verifier outputs 1 indicating accepting the proof that  $\langle \mathbf{f}, \mathbf{q} \rangle = v$ , or outputs 0 indicating rejecting the proof.

**Definition 21 (Completeness).** For all  $l \leq D$ , for all inputs  $f_0, \dots, f_{l-1} \in \mathbb{F}$ , for query vectors  $\mathbf{q} \in \mathbb{F}^l$ ,

$$\Pr \left( b = 1 : \begin{array}{l} \text{pp} \leftarrow \mathbf{Setup}(1^\kappa, D) \\ (C, \tilde{\mathbf{c}}) \leftarrow \mathbf{Com}(\text{pp}, f_0, \dots, f_{l-1}, l) \\ v \leftarrow \langle (f_0, \dots, f_{l-1}), \mathbf{q} \rangle \\ b \leftarrow \mathbf{Eval}(\text{pp}, C, l, \mathbf{q}, v; \mathbf{f}) \end{array} \right) = 1.$$

**Definition 22 (Binding).** An Inner Product Commitment scheme  $\text{PC}$  is binding if for all PPT  $\mathcal{A}$ , the following probability is negligible in  $\kappa$ .

$$\Pr \left( \begin{array}{l} \mathbf{Open}(\text{pp}, \mathbf{f}_0, l, C, \tilde{\mathbf{c}}_0) = 1 \wedge \\ \mathbf{Open}(\text{pp}, \mathbf{f}_1, l, C, \tilde{\mathbf{c}}_1) = 1 \wedge \\ \tilde{\mathbf{c}}_0 \neq \tilde{\mathbf{c}}_1 \end{array} : \begin{array}{l} \text{pp} \leftarrow \text{setup}(1^\kappa, D) \\ (C, \mathbf{f}_0, \mathbf{f}_1, \tilde{\mathbf{c}}_0, \tilde{\mathbf{c}}_1, l) \leftarrow \mathcal{A}(\text{pp}) \end{array} \right).$$

**Definition 23 (Succinctness).** We require the commitments and the evaluation proofs to be of size independent of the length of the vector, that is the scheme is proof succinct if  $|C|$  is  $\text{poly}(\kappa)$  and  $|\pi|$  is  $\text{poly}(\kappa)$ , where  $\pi$  is the transcript obtained by applying FS to  $\mathbf{Eval}$ .

**Definition 24 (Extractability).** For any PPT adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ , there exists a PPT algorithm  $\text{Ext}$  such that the following probability is negligible in  $\kappa$ :

$$\Pr \left( b = 1 \wedge \mathcal{R}_{\text{Eval}}(\text{pp}, C, l, \mathbf{q}, v; \mathbf{f}, \tilde{\mathbf{c}}) = 0 : \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\kappa, D) \\ (C, l, \mathbf{q}, v, \text{st}) \leftarrow \mathcal{A}_1(\text{pp}) \\ (\mathbf{f}, \tilde{\mathbf{c}}) = \text{Ext}^{\mathcal{A}_2}(\text{pp}) \\ b \leftarrow \langle \mathcal{A}_2(\text{st}), V_{\text{Eval}} \rangle(\text{pp}, C, l, \mathbf{q}, v) \end{array} \right).$$

where the relation  $\mathcal{R}_{\text{Eval}}$  is defined as follows:

$$\mathcal{R}_{\text{Eval}} = \{ ((\text{pp}, C \in \mathbb{G}, l \in \mathbb{N}, \mathbf{q} \in \mathbb{F}^l, v \in \mathbb{F}); (\mathbf{f}, \tilde{\mathbf{c}})) : (\text{Open}(\text{pp}, \mathbf{f}, l, C, \tilde{\mathbf{c}}) = 1) \wedge v = \langle \mathbf{f}, \mathbf{q} \rangle \pmod{p} \}$$

### 2.2 Polynomial Commitment Scheme

The notion of a polynomial commitment scheme that allows the prover to open evaluations of the committed polynomial succinctly was introduced in [18] who gave a construction under the trusted setup assumption. A polynomial commitment scheme over  $\mathbb{F}$  is a tuple  $\text{PC} = (\text{setup}, \text{commit}, \text{open}, \text{eval})$  where:

- $\text{setup}(1^\kappa, D) \rightarrow \text{pp}$ . On input security parameter  $\kappa$ , and an upper bound  $D \in \mathbb{N}$  on the degree,  $\text{setup}$  generates public parameters  $\text{pp}$ .
- $\text{commit}(\text{pp}, f(X), d) \rightarrow (C, \tilde{\mathbf{c}})$ . On input the public parameters  $\text{pp}$ , and a univariate polynomial  $f(X) \in \mathbb{F}[X]$  with degree at most  $d \leq D$ ,  $\text{commit}$  outputs a commitment to the polynomial  $C$ , and additionally an opening hint  $\tilde{\mathbf{c}}$ .
- $\text{open}(\text{pp}, f(X), d, C, \tilde{\mathbf{c}}) \rightarrow b$ . On input the public parameters  $\text{pp}$ , the commitment  $C$  and the opening hint  $\tilde{\mathbf{c}}$ , a polynomial  $f(X)$  of degree  $d \leq D$ ,  $\text{open}$  outputs a bit indicating accept or reject.
- $\text{eval}(\text{pp}, C, d, x, v; f(X)) \rightarrow b$ . A public coin interactive protocol  $\langle P_{\text{eval}}(f(X)), V_{\text{eval}} \rangle(\text{pp}, C, d, z, v)$  between a PPT prover and a PPT verifier. The parties have as common input public parameters  $\text{pp}$ , commitment  $C$ , degree  $d$ , evaluation point  $x$ , and claimed evaluation  $v$ . The prover has, in addition, the opening  $f(X)$  of  $C$ , with  $\text{deg}(f) \leq d$ . At the end of the protocol, the verifier outputs 1 indicating accepting the proof that  $f(x) = v$ , or outputs 0 indicating rejecting the proof.

A polynomial commitment scheme must satisfy completeness, binding and extractability.

**Definition 25 (Completeness).** For all polynomials  $f(X) \in \mathbb{F}[X]$  of degree  $d \leq D$ , for all  $x \in \mathbb{F}$ ,

$$\Pr \left( b = 1 : \begin{array}{l} \text{pp} \leftarrow \text{setup}(1^\kappa, D) \\ (C, \tilde{\mathbf{c}}) \leftarrow \text{commit}(\text{pp}, f(X), d) \\ v \leftarrow f(x) \\ b \leftarrow \text{eval}(\text{pp}, C, d, x, v; f(X)) \end{array} \right) = 1.$$

**Definition 26 (Binding).** A polynomial commitment scheme  $\text{PC}$  is binding if for all PPT  $\mathcal{A}$ , the following probability is negligible in  $\kappa$ :

$$\Pr \left( \begin{array}{l} \text{open}(\text{pp}, f_0, d, C, \tilde{\mathbf{c}}_0) = 1 \wedge \\ \text{open}(\text{pp}, f_1, d, C, \tilde{\mathbf{c}}_1) = 1 \wedge \\ f_0 \neq f_1 \end{array} : \begin{array}{l} \text{pp} \leftarrow \text{setup}(1^\kappa, D) \\ (C, f_0, f_1, \tilde{\mathbf{c}}_0, \tilde{\mathbf{c}}_1, d) \leftarrow \mathcal{A}(\text{pp}) \end{array} \right).$$

**Definition 27 (Extractability).** For any PPT adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ , there exists a PPT algorithm  $\text{Ext}$  such that the following probability is negligible in  $\kappa$ :

$$\Pr \left( \begin{array}{l} b = 1 \wedge \mathcal{R}_{\text{eval}}(\text{pp}, C, x, v; \tilde{f}, \tilde{\mathbf{c}}) = 0 : \\ \text{pp} \leftarrow \text{setup}(1^\kappa, D) \\ (C, d, x, v, \text{st}) \leftarrow \mathcal{A}_1(\text{pp}) \\ (\tilde{f}, \tilde{\mathbf{c}}) \leftarrow \text{Ext}^{\mathcal{A}_2}(\text{pp}) \\ b \leftarrow \langle \mathcal{A}_2(\text{st}), V_{\text{eval}} \rangle(\text{pp}, C, d, x, v) \end{array} \right).$$

where the relation  $\mathcal{R}_{\text{eval}}$  is defined as follows:

$$\mathcal{R}_{\text{eval}} = \{ ((\text{pp}, C \in \mathbb{G}, x \in \mathbb{F}, v \in \mathbb{F}); (f(X), \tilde{\mathbf{c}})) : (\text{open}(\text{pp}, f, d, C, \tilde{\mathbf{c}}) = 1) \wedge v = f(x) \}$$

**Definition 28 (Succinctness).** We require the commitments and the evaluation proofs to be of size independent of the degree of the polynomial, that is the scheme is proof succinct if  $|C|$  is  $\text{poly}(\kappa)$ ,  $|\pi|$  is  $\text{poly}(\kappa)$  where  $\pi$  is the transcript obtained by applying FS to eval. Additionally, the scheme is verifier succinct if eval runs in time  $\text{poly}(\kappa) \cdot \log(d)$  for the verifier.

### 2.3 Assumptions

*Groups of Unknown Order and GGM.* Our constructions make use of groups of unknown order. A class group is a candidate group of unknown order. The *class group* of an imaginary quadratic order [6, 7] is the quotient group of fractional ideals by principal ideals of an order of a number field with ideal multiplication. It is completely defined by its discriminant, which can be generated using only public randomness.

We use the generic group model (GGM) for groups of unknown order as defined by Damgård and Koprowski [14], and used in [5]. In this model, the group is parameterized by two integer public parameters  $A, B$  and the order of the group is sampled uniformly from  $[A, B]$ . The group  $\mathbb{G}$  description consists of a random injective function  $\sigma : \mathbb{Z}_{|\mathbb{G}|} \rightarrow \{0, 1\}^\ell$ , for some  $\ell$  where  $2^\ell \gg |\mathbb{G}|$ . The elements of the group are  $\sigma(0), \sigma(1), \dots, \sigma(|\mathbb{G}| - 1)$ . A generic group algorithm  $\mathcal{A}$  is a probabilistic algorithm with the following properties. Let  $\mathcal{L}$  be a list that is initialized with the encodings (group elements) given to  $\mathcal{A}$  as inputs.  $\mathcal{A}$  can query two generic group oracles,  $\mathcal{O}_1$  and  $\mathcal{O}_2$ .  $\mathcal{O}_1$  samples a random  $r \in \mathbb{Z}_{|\mathbb{G}|}$  and returns  $\sigma(r)$  which is appended to  $\mathcal{L}$ . The second oracle  $\mathcal{O}_2(i, j, \pm)$  takes two indices  $i, j \in \{1, \dots, q\}$ , where  $q$  is the size of  $\mathcal{L}$ , and a sign bit and returns  $\sigma(x_i \pm x_j)$ , which is appended to  $\mathcal{L}$ . It should be noted that  $\mathcal{A}$  is not given  $|\mathbb{G}|$ .

We use a group sampler  $\mathsf{GGen}$  that on input a security parameter  $\kappa$ , samples a description of the group  $\mathbb{G}$  of size  $2^{\text{poly}(\kappa)}$ . Note that  $\mathsf{GGen}$  is public-coin.

We informally describe the rest of the assumptions – note that these problems are indeed intractable in the GGM. The formal definitions are deferred to the full version.

*Adaptive root assumption.* Computing random roots of arbitrary group elements  $g$  is hard for any PPT adversary.

*Low order assumption.* Computing the order of any non-trivial element in a group  $\mathbb{G} \leftarrow \mathsf{GGen}$  is hard for any PPT adversary.

## 2.4 Proofs about Exponents

**PoE** (*Proof of Exponentiation*): We use Wesolowski’s proof of exponentiation (PoE) protocol [26] in its slightly more generalized form as presented in [5] for the relation  $\mathcal{R}_{\text{PoE}} = \{(u, w \in \mathbb{G}, x \in \mathbb{Z}; \perp) : w = u^x \in \mathbb{G}\}$ .

**PoKE** (*Proof of Knowledge of Exponent*): We also use the PoKE protocol from [5] in our protocols. This protocol is an argument of knowledge in the GGM for the relation  $\mathcal{R}_{\text{PoKE}} = \{(u, w \in \mathbb{G}; x \in \mathbb{Z}) : w = u^x \in \mathbb{G}\}$ .

**PoKPE** (*Proof of Knowledge of Positive exponent*): Define the relation  $\mathcal{R}_{\text{PoKPE}} = \{(w \in \mathbb{G}; x \in \mathbb{Z}) : (w = g^x) \wedge (x > 0)\}$ . We construct an argument of knowledge for this relation called PoKPE using PoKE and Lagrange’s four-square theorem. We also use the notation  $\text{PoKPE}\{A, B, \dots\}$  to denote the combined protocol for the set, where the verifier outputs 1 iff PoKPE checks pass for all elements. More details about these protocols appear in the full version.

## 3 Inner Product Commitment Scheme with Constant-Sized Proof

In this section, we construct an inner product commitment (IPC) scheme that achieves constant-sized proof and linear time verification.

### 3.1 Construction

IPC = (**Setup**, **Com**, **Open**, **Eval**) are as defined below:

- **Setup**( $1^\kappa, D$ ): Here,  $\kappa$  is the security parameter and  $D$  is an upper bound on the length of the committed vectors. Sample a group of unknown order (we use class groups)  $\mathbb{G} \leftarrow \mathsf{GGen}(\kappa)$  and  $g \leftarrow_{\$} \mathbb{G}$ . Define  $\alpha = p^{2D}$  ( $p$  is a large prime such that  $\text{len}(p) = \text{poly}(\kappa)$ ). Return  $\text{pp} = (\kappa, \mathbb{G}, g, p)$  ( $\alpha$  does not have to be explicitly returned; it is defined completely by  $p, D$ ).

- **Com**(pp,  $D, f_0, \dots, f_{l-1}, l$ ): Define the commitment to  $\mathbf{f} = (f_0, \dots, f_{l-1}) \in \mathbb{Z}_p^l$  as  $C := g^{\sum_{i=0}^{l-1} f_i \alpha^{2i}}$ , considering  $f_i \in \mathbb{Z}_p$  as integers in  $[0, p - 1]$  and the sum in  $\mathbb{Z}$ . If  $l \leq D$ , return  $(C, \mathbf{f})$ , else return error.
- **Open**(pp,  $\mathbf{f}, l, C, \tilde{\mathbf{c}}$ ): ( $\mathbf{f}$  is the claimed opening and  $\tilde{\mathbf{c}}$  is an opening hint) Return 1 if all the below conditions hold, else return 0.
  - $l \leq D, \tilde{\mathbf{c}} = \mathbf{f} \pmod p$  <sup>6</sup>
  - $C = g^{\sum_{i=0}^{l-1} \tilde{c}_i \alpha^{2i}}$ , exponent  $\sum_{i=0}^d \tilde{c}_i \alpha^{2i} \in \mathbb{Z}$  and  $\tilde{\mathbf{c}} \in \mathbb{Q}(2, 3)^l$ , where

$$\mathbb{Q}(\beta_1, \beta_2) := \left\{ \frac{a}{b} : \gcd(b, p) = 1, 0 < b < p^{\beta_1}, |a/b| \leq \beta_2 \alpha \right\},$$

where  $|a|$  denote s the absolute value of  $a \in \mathbb{Q}$ . (Note that  $\mathbb{Q}(\beta_1, \beta_2)$  is a subset of  $\mathbb{Q}(\beta'_1, \beta'_2)$  if  $\beta_1 \leq \beta'_1, \beta_2 \leq \beta'_2$ )

- **Eval**(pp,  $C, l, \mathbf{q}, v; \mathbf{f}$ ): The **Eval** protocol consists of two sub-protocols **TEST** and **IPP** as described in Fig. 2 and 3 below.
  - $b_1 \leftarrow \mathbf{TEST}(C, l; \mathbf{f}), b_2 \leftarrow \mathbf{IPP}(C, l, \mathbf{q}, v; \mathbf{f})$ . Return  $b = (l \leq D) \wedge b_1 \wedge b_2$

### 3.2 Proofs of Security

We prove that our construction IPC satisfies the requirements of an inner product scheme as defined in §2.1.

**Theorem 31 (Completeness).** *The inner product commitment scheme IPC satisfies Completeness (Definition 21).*

*Proof.* Note that by definition of **CoeffSplit** and completeness of **PoKPE**, all the **PoKPE** checks will accept.

To show that the last checks in **TEST** and **IPP** hold, it suffices to show that  $v = 0$  in **TEST** and  $v = \langle \mathbf{f}, \mathbf{q} \rangle \pmod p$  in **IPP**. We will show this by expanding the computations done in **CoeffSplit**.

In **TEST**, direct manipulation shows

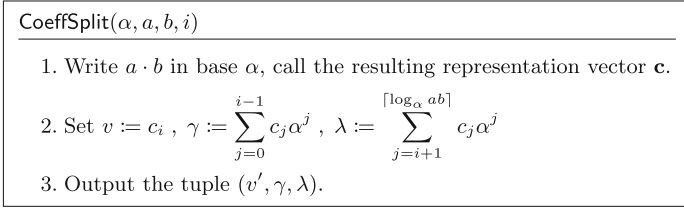
$$\begin{aligned} & \sum_{j=0}^{l-1} f_j \alpha^{2j} \times \sum_{j=0}^{l-1} \alpha^{2l-2-2j} z_j \\ &= \alpha^{2l} \underbrace{\left( \sum_{j' > j} \alpha^{2(j'-j)-2} f_{j'} z_j \right)}_{\lambda} + \underbrace{\left( \sum_{j' < j} \alpha^{2l-2-2(j-j')} f_{j'} z_j + \sum_{j'=j} \alpha^{2l-2} f_j z_j \right)}_{\gamma} \end{aligned}$$

and notice that since  $\alpha > lp^2$ , these are indeed the  $\gamma, \lambda$  returned by **CoeffSplit**, and  $v = 0$  (Fig. 1).

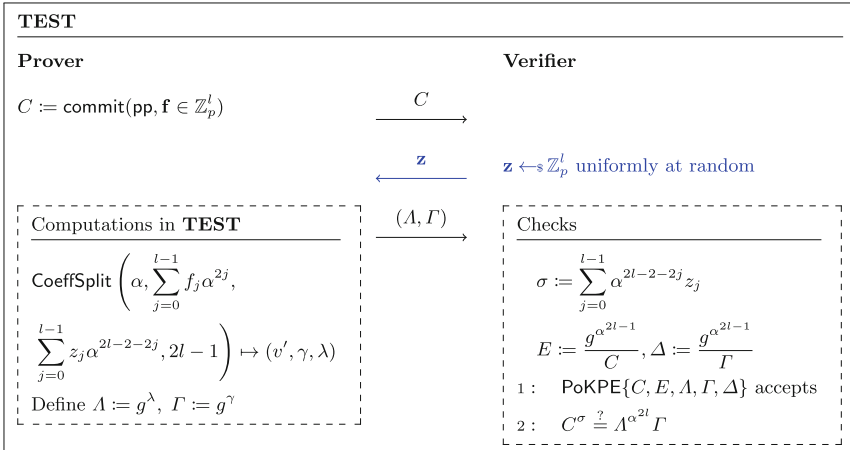
---

<sup>6</sup> Treating any coordinate  $\frac{a}{b}$  of  $\tilde{\mathbf{c}} \pmod p$  as  $a' \cdot b'^{-1}$ , where  $a' = (a \pmod p) \in \mathbb{Z}_p$  and  $b' = (b \pmod p) \in \mathbb{Z}_p$ .



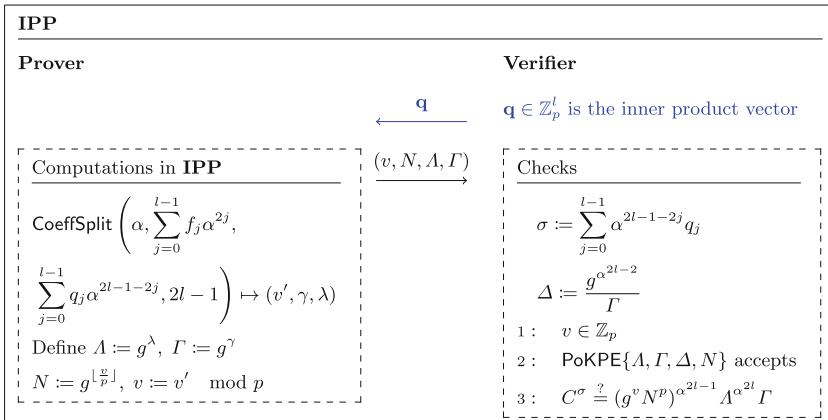


**Fig. 1.** CoeffSplit



The blue colored parts will be replaced in subsequent versions of this protocol.

**Fig. 2.** The TEST Protocol



The blue colored parts will be replaced in subsequent versions of this protocol.

**Fig. 3.** The IPP Protocol

And, in **IPP**,

$$\begin{aligned} \sum_{j=0}^{l-1} f_j \alpha^{2j} \times \sum_{j=0}^{l-1} \alpha^{2l-1-2j} q_j &= \alpha^{2l-1} \left( \underbrace{\sum_{i=0}^{l-1} f_i q_i}_{v} \bmod p + p \underbrace{\left\lfloor \frac{\sum_{i=0}^{l-1} f_i q_i}{p} \right\rfloor}_n \right) \\ &+ \underbrace{\alpha^{2l-2}(0) + \alpha^{2l} \left( \sum_{j' > j} \alpha^{2(j'-j)-1} f_{j'} q_j \right)}_{\lambda} + \underbrace{\left( \sum_{j' < j} \alpha^{2l-1-2(j-j')} f_{j'} q_j \right)}_{\gamma} \end{aligned}$$

Here, again since  $\alpha > lp^2$ ,  $(v + np, \lambda, \gamma)$  above coincide with the output of **CoeffSplit**, and  $v = \langle \mathbf{f}, \mathbf{q} \rangle \bmod p$ .

**Theorem 32 (Binding).** *The inner product commitment scheme IPC Construction in Sect. 3.1 is binding (Definition 22) for opening hint vectors in  $\mathbb{Q}(\beta_1, \beta_2)$  as long as  $\alpha > 4\beta_2 p^{2\beta_1}$  and if the Order assumption holds for **GGen**.*

*Proof.* Suppose there exists an adversary  $\mathcal{A}$  which breaks binding as defined in Definition 22, i.e.,  $\mathcal{A}(\text{pp})$  outputs  $(C, \mathbf{f}, \mathbf{f}', \mathbf{c}, \mathbf{c}', d)$  such that  $\text{open}(\text{pp}, \mathbf{f}, d, C, \mathbf{c}) = 1$  and  $\text{open}(\text{pp}, \mathbf{f}', d, C, \mathbf{c}') = 1$  but  $\mathbf{f} \neq \mathbf{f}'$  (which also implies that  $\mathbf{c} \neq \mathbf{c}'$  – we will use this condition to show a contradiction).

Then, since  $\text{open}$  outputs 1 for both  $\mathbf{f}, \mathbf{f}'$  we know that the opening hints  $\mathbf{c}, \mathbf{c}' \in \mathbb{Q}(\beta_1, \beta_2)^l$  and that  $g^{\sum_{i=0}^{l-1} c_i \alpha^{2i}} = g^{\sum_{i=0}^{l-1} c'_i \alpha^{2i}} \iff g^{\sum_{i=0}^{l-1} (c_i - c'_i) \alpha^{2i}} = 1$ . If the exponent of  $g$  above were not zero, we could construct an adversary  $\mathcal{A}_{Ord}$  that uses the above exponent to break the Low order assumption. Now, let the exponents be equal, and consider the largest index  $j$  such that  $c'_i \neq c_i$  (WLOG, let  $c'_i > c_i$ ). This implies that  $\sum_{i=0}^{j-1} (c_i - c'_i) \alpha^{2i} = (c'_j - c_j) \alpha^{2j}$ .

We can now show that this equality is impossible given the conditions on  $\alpha, \beta_1, \beta_2$ .

Notice that any difference  $|c'_i - c_i|$  (if non-zero) can be bounded by  $\frac{1}{p^{2\beta_1}} < |c'_i - c_i| < 2\beta_2 \alpha$ , since  $c_i, c'_i \in \mathbb{Q}(\beta_1, \beta_2)$ . This gives a contradiction, since

$$\sum_{i=0}^{j-1} (c_i - c'_i) \alpha^{2i} < 2\beta_2 \alpha \sum_{i=0}^{j-1} \alpha^{2i} < 2\beta_2 \alpha \cdot 2\alpha^{2j-2} < \frac{\alpha^{2j}}{p^{2\beta_1}} < (c'_j - c_j) \alpha^{2j},$$

Before proving extractability, we need a few definitions. Define

$$S := \left\{ \frac{m\alpha - n}{k} : m, n, k \in \mathbb{Z}, \gcd(m, k) = 1, 0 < m \leq k < p, -2 < n < k + 2 \right\}$$

as a subset of  $\mathbb{Z}$  and functions  $\chi_m, \chi_n : S^q \rightarrow \mathbb{Q}^q$  which isolates the vector of fractions  $m/k$  and  $n/k$  from the elements of  $S^q$ :

$$\mathbf{v} \in S^q \implies \mathbf{v} = \left( \frac{m_i \alpha - n_i}{k_i} \right)_i, \quad \chi_m(\mathbf{v}) := \left( \frac{m_i}{k_i} \right)_i, \quad \text{and} \quad \chi_n(\mathbf{v}) := \left( \frac{n_i}{k_i} \right)_i.$$

These functions can be made well-defined by fixing a representation of elements of  $S$ : for any  $d \in S$ , consider the representation  $(m, n, k)$  as the one with the smallest denominator  $k$  and if there are multiple such representations, we pick the one with the smallest  $m$ .

**Theorem 33 (Extractability).** *The inner product commitment scheme IPC satisfies Extractability for  $(\beta_1, \beta_2) = (2, 3)$  (Definition 24) in the Generic Group Model.*

*Proof.* We split the proof into two theorems; the first theorem (concerning **TEST**) will define a partial extractor and obtain conditions on the extracted objects, while the second theorem uses the results and the extractor of the first theorem and finishes the proof.

Suppose there exists a *generic* adversary  $\mathcal{A}$  that makes the Verifier in **eval** accept with non-negligible probability (and hence both the **TEST** verifier and **IPP** verifier). We will construct a polynomial time extractor **Ext** that outputs  $(\tilde{\mathbf{f}}, \tilde{\mathbf{c}})$  satisfying  $\mathcal{R}_{\text{eval}}$  (in Definition 27) with overwhelming probability.

**Theorem 34 (TEST Extractor).** *If the Verifier in **TEST** outputs accept with non-negligible probability, there exists an efficient extractor  $\text{Ext}_T$  in the GGM that outputs  $\mathbf{c}, \mathbf{d} \in [\alpha]^l$  such that  $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$  and  $\mathbf{d} \in S^l$ .*

**Theorem 35 (IPP Extractor).** *If the Verifier in **IPP** outputs accept with non-negligible probability (and given that the Verifier of **TEST** also did so), there exists an efficient extractor **Ext** in the GGM that outputs an opening  $\mathbf{f} \in \mathbb{Z}_p^l$  and opening hint  $\tilde{\mathbf{c}} \in \mathbb{Q}(2, 3)$  for  $C$  such that  $v = \langle \tilde{\mathbf{f}}, \mathbf{q} \rangle \pmod p$  and satisfies Extractability (Definition 24).*

### 3.3 Auxiliary Lemmas

**Lemma 1.** *Suppose  $K = \sum_{i=0}^k M_i \alpha^i$  where  $M_i$ 's are not necessarily  $< \alpha$ , but we have a bound  $M_i < \alpha(\alpha - 1) \forall i$ . Then, we can write  $K = \sum_{i=0}^{k+1} U_i \alpha^i$  where each  $U_i < \alpha$ ,  $u_i \in \{0, 1\}$ , and*

$$U_i := \begin{cases} (M_0 \pmod{\alpha + u_0}) \pmod{\alpha} & \text{if } i = 0 \\ (M_i \pmod{\alpha + \lfloor \frac{M_{i-1}}{\alpha} \rfloor + u_i}) \pmod{\alpha} & \text{if } 1 \leq i \leq k \\ (\lfloor \frac{M_k}{\alpha} \rfloor + u_{k+1}) & \text{if } i = k + 1 \end{cases}$$

$$u_i := \begin{cases} 0 & \text{if } i = 0, 1 \\ \left\lfloor \frac{M_{i-1} \pmod{\alpha + \lfloor \frac{M_{i-2}}{\alpha} \rfloor + u_{i-1}}}{\alpha} \right\rfloor & \text{if } 1 \leq i \leq k + 1 \end{cases}$$

**Lemma 2.** *Suppose for some  $\alpha$ ,  $M'\alpha - N' = M\alpha - N$ , where  $M', N' \in \mathbb{Q}$  and  $M, N \in \mathbb{Z}$ . If  $|N|, |N'| < B$ ,  $M' = \frac{x}{y}$  and  $y < \frac{\alpha}{2B}$ , then  $M' = M$  and  $N' = N$ .*

### 3.4 Proof of Theorem 34 - TEST Extraction

First, we prove a lemma giving a partial extractor for the **TEST** protocol.

**Lemma 3.** *If the Verifier in **TEST** outputs accept with non-negligible probability, there exists an efficient extractor  $\text{Ext}_T$  in the GGM that outputs with high probability  $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$  (that only depends on the commitment  $C$ ) such that  $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$ .*

*Moreover, if the random vector used in **TEST** was  $\mathbf{z} \in \mathbb{Z}_p^l$ , then we also have the relation (for some  $u \in \{0, 1\}$ ):*

$$\langle \mathbf{d}, \mathbf{z} \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u = 0 \pmod{\alpha}$$

*Proof.* We define an extractor  $\text{Ext}_T$  that invokes the PoKPE extractor for  $C$ , which outputs an exponent  $c > 0$  such that  $C = g^c$ . Since  $E$  also passes the PoKPE protocol and  $C \cdot E = g^{\alpha^{2l-1}}$ , we can infer that  $c < \alpha^{2l-1}$ . Consider the base- $\alpha$  representation of  $c$ , which is a  $2l$ -length vector.  $\text{Ext}_T$  outputs the even indexed coordinates as  $\mathbf{c}$  and the odd indexed coordinates as  $\mathbf{d}$ . By construction,  $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$ .

Notice that each PoKPE is essentially a range check as well as a proof of knowledge of the exponent. With overwhelming probability, we can assume that each of these statements are true (due to knowledge soundness of PoKPE). Hence, we get that the prover “knows” (formally, an extractor outputs) integers  $c, \gamma, \lambda$  such that  $C = g^c$ ,  $0 < c < \alpha^{2l-1}$ ,  $\Gamma = g^\gamma$ ,  $0 < \gamma < \alpha^{2l-1}$ , and  $\Lambda = g^\lambda$ ,  $0 < \lambda$ .

Now, notice that for any equality of group elements in a group of unknown order, we can (with overwhelming probability) equate their exponents when written with base  $g$  over integers. This follows from the Low order assumption as long as the prover knows all the exponents w.r.t. some fixed base  $g$  (else the prover could compute a multiple of the order of the group).

Hence, given an equation of the form  $C^\sigma = \Gamma g^{v\alpha^{2l-1}} \Lambda^{\alpha^{2l}}$  (this is essentially Check 2 for the **TEST** verifier with  $y = 0$ ), we can write

$$\begin{aligned} g^{c\sigma} &= g^{\gamma + v\alpha^{2l-1} + \lambda\alpha^{2l}} \\ \implies c\sigma &= \gamma + v\alpha^{2l-1} + \lambda\alpha^{2l} \end{aligned}$$

Writing this in base  $\alpha$  and using Lemma 1, we can compare the coefficients of  $\alpha^{2l-1}$  on both sides:

$$v = \langle \mathbf{d}, \mathbf{z} \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u \pmod{\alpha}$$

In **TEST**, we have  $v = 0$ , hence we prove the lemma.

Now, using the above lemma, we show that if the prover succeeds with non-negligible probability, we must have  $d_i \in S$  for all  $i$ .

If the prover succeeds with non-negligible probability, it must hold that it also succeeds for a non-negligible probability over the choice of the random query  $\mathbf{z} \in \mathbb{Z}_p^l$ . Fix some index  $0 \leq i \leq l-1$ .

Partition the randomness space  $\mathbb{Z}_p^l$  into 1-dimensional “lines” of length  $p$  along the  $i^{\text{th}}$  dimension:

$$T_{\mathbf{q}} := \{(z_i, \mathbf{q}) : z_i \in \mathbb{Z}_p\}$$

If the prover was only able to succeed for at most one value of  $\mathbf{z}$  in all  $T_{\mathbf{q}}$ , the overall success probability of the prover is bounded by  $\frac{1}{p}$ , which is negligible and hence a contradiction. Hence, there must exist some  $\mathbf{q}$  such that there exists two points  $\mathbf{z}_1, \mathbf{z}_2 \in T_{\mathbf{q}}$  such that the prover succeeds in convincing the verifier (for simplicity, we will use  $z_1, z_2$  to denote the  $i^{\text{th}}$  coordinate that differs in these two vectors. WLOG let  $z_2 > z_1$ ).

Now, using Lemma 3, we get two equations:

$$\langle \mathbf{d}, \mathbf{z}_1 \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}_1 \rangle}{\alpha} \right\rfloor + u_1 \pmod{\alpha} = 0 \pmod{\alpha} \tag{6}$$

$$\langle \mathbf{d}, \mathbf{z}_2 \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}_2 \rangle}{\alpha} \right\rfloor + u_2 \pmod{\alpha} = 0 \pmod{\alpha} \tag{7}$$

where  $u_1, u_2 \in \{0, 1\}$ . Our aim is to isolate and prove conditions on a single coordinate  $d_i$ , and notice that the inner products  $\langle \mathbf{d}, \mathbf{z}_1 \rangle$  and  $\langle \mathbf{d}, \mathbf{z}_2 \rangle$  differ only in the  $i^{\text{th}}$  term. Hence, subtracting the two equations, we get:

$$(z_2 - z_1)d_i = - \left( \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}_2 \rangle}{\alpha} \right\rfloor - \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z}_1 \rangle}{\alpha} \right\rfloor + u_2 - u_1 \right) \pmod{\alpha}$$

Call the term in the brackets on the RHS  $n$ . Using the fact that  $x - 1 \leq \lfloor x \rfloor < x$  and  $u_i \in \{0, 1\}$  for all  $i$  gives us trivial bounds  $-2 < n < (z_2 - z_1) + 2$ . We also know that  $0 < z_2 - z_1 < p$ . Letting  $k := z_2 - z_1$ ,

$$kd_i = -n \pmod{\alpha} \implies d_i = \frac{m\alpha - n}{k}$$

where  $-2 < n < k + 2, 0 < k < p$  and  $0 < m \leq k$  since  $d_i < \alpha$ .

Hence  $d_i \in S$ . Since  $i$  was an arbitrary index,  $\mathbf{d} \in S^l$ .

### 3.5 Proof of Theorem 35 – IPP Extraction

We define the final extractor  $\text{Ext}$  for  $\text{eval}$  using the extractor  $\text{Ext}_T$  from **TEST** that outputs  $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_p^l$ . Specifically,  $\text{Ext}$  invokes  $\text{Ext}_T$  and performs the following additional computations on  $\mathbf{c}, \mathbf{d}$ :

1. Compute  $m_i, n_i, k_i$  for every  $i$  such that  $d_i = \frac{m_i\alpha - n_i}{k_i}$ .
2. Let  $m_{-1} := 0, k_{-1} := 1$  and define vectors  $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_p^l$  as  $c'_i := c_i + \frac{m_i - 1}{k_i - 1}$  and  $d'_i := -\frac{n_i}{k_i}$ .
3. Output  $\mathbf{c}' + \alpha\mathbf{d}'$  as the opening hint, and  $(\mathbf{c}' + \alpha\mathbf{d}') \pmod{p}$  as the opening to the commitment.

Note that this extractor is indeed efficient as  $\text{Ext}_T$  is efficient and the only non-trivial computations done are in Step 1 above, which can be done efficiently (details in full version).

By construction, this is a valid opening hint, as  $\mathbf{d} \in S^l \implies \mathbf{c}' + \alpha \mathbf{d}' \in \mathbb{Q}(2, 3)$ . This is just a rearrangement of the coordinates of  $\mathbf{c}$  and  $\mathbf{d}$  and keeps the sum  $\sum_{i=0}^{l-1} (c'_i + \alpha d'_i) \alpha^{2i}$  equal to the previous sum  $\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}$  (since we just move the coefficient of  $\alpha$  in  $d_i$  to  $c_{i+1}$ ). Hence,  $C = g^{\sum_{i=0}^{l-1} (c'_i + \alpha d'_i) \alpha^{2i}}$  and the exponent  $\in \mathbb{Z}$ .

Now, since the verifier accepts in **IPP**, we can use similar arguments as made in Lemma 3 for the check equation in **IPP** to get

$$c\sigma = \gamma + 0\alpha^{2l-2} + (v + np)\alpha^{2l-1} + \lambda\alpha^{2l}$$

Focusing on the coefficients of  $\alpha^{2l-2}$  and  $\alpha^{2l-1}$ , we get two equations:

$$\langle \mathbf{d}, \mathbf{q}^+ \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u' = 0 \pmod{\alpha} \quad (8)$$

$$\langle \mathbf{c}, \mathbf{q} \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u = v + np \pmod{\alpha} \quad (9)$$

where  $\mathbf{q}^+$  is defined as the vector with elements  $q_i^+ := q_{i+1} \forall i \in \{0, \dots, l-2\}$ ,  $q_{l-1}^+ := 0$  and  $u, u' \in \{0, 1\}$ .

Due to the bounds on coefficients of  $\mathbf{q}$  (chosen by the verifier  $\in \mathbb{Z}_p$ ), we know that Eq. 8's LHS over integers must be either 0 or  $\alpha$ . Also define

$$M' := \sum_{i=0}^{l-1} \frac{m_i}{k_i} q_i^+, \quad \text{and} \quad N' := \sum_{i=0}^{l-1} \frac{n_i}{k_i} q_i^+$$

1. If the LHS is 0, then so are each of the terms in the LHS, as they are all non-negative. Hence,  $\langle \mathbf{d}, \mathbf{q}^+ \rangle = 0 \pmod{\alpha}$  which implies  $u = 0$  (due to Lemma 1). Hence, we can simplify Eq. 9

$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \pmod{\alpha} \\ &= \langle \mathbf{c}, \mathbf{q} \rangle + \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \pmod{\alpha} = \langle \mathbf{c}, \mathbf{q} \rangle + M' - \frac{N'}{\alpha} \pmod{\alpha} \end{aligned}$$

Since  $M' - N'/\alpha$  must be an integer and  $N' < \alpha \implies N' = 0$ .

$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle + M' \pmod{\alpha} = \langle \mathbf{c}', \mathbf{q} \rangle \pmod{\alpha} \\ \implies v &= \langle \mathbf{c}', \mathbf{q} \rangle \pmod{p} = \langle \mathbf{c}', \mathbf{q} \rangle + \alpha \langle \mathbf{d}', \mathbf{q} \rangle \pmod{p}, \end{aligned}$$

as  $\alpha = 0 \pmod{p}$  and  $\langle \mathbf{d}', \mathbf{q} \rangle$  is invertible modulo  $p$  (or simply  $0 \pmod{p}$ ). In either case,  $v = \langle \mathbf{c}' + \alpha \mathbf{d}', \mathbf{q} \rangle \pmod{p}$ .

2. If the LHS is  $\alpha$ , we get  $u = 1$ . Now, write Eq. 8 in the form  $\langle \mathbf{d}, \mathbf{q}^+ \rangle = M\alpha - N$  by moving all the terms but the inner product to the RHS and calling it

$N$ . Now, we get  $M'\alpha - N' = M\alpha - N$  where  $|N|, |N'| < 3pl$  and  $M'$  has denominator at most the LCM of all the  $k_i$ , which is at most  $p^l$ . We can apply Lemma 2 which implies that  $M', N' \in \mathbb{Z}$  (since  $\alpha > p^{2l}$ ). Then,

$$\begin{aligned} v + np &= \langle \mathbf{c}, \mathbf{q} \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{d}, \mathbf{q}^+ \rangle}{\alpha} \right\rfloor + u \pmod{\alpha} \\ &= \langle \mathbf{c}, \mathbf{q} \rangle \pmod{\alpha} + \left\lfloor M' - \frac{N'}{\alpha} \right\rfloor + 1 \pmod{\alpha} \\ &= \langle \mathbf{c}, \mathbf{q} \rangle \pmod{\alpha} + M' - 1 + 1 \pmod{\alpha} \end{aligned}$$

as  $N' < \alpha$ . Hence, as before, we get that  $v = \langle \mathbf{c}' + \alpha \mathbf{d}', \mathbf{q} \rangle \pmod{p}$ .

Thus, the extracted opening equals the claimed inner product  $v$  in both cases and we satisfy extractability.

**Non-interactivity Using Fiat-Shamir.** Protocol `eval` is public-coin and we can use the Fiat-Shamir heuristic [16] to obtain a non-interactive version in the ROM that has a constant-sized proof. The prover applies the RO on the commitment  $C$  to obtain the random query vector  $\mathbf{z}$ . Note that, the query vector  $\mathbf{q}$  itself needs to be communicated, but the size of the proof is constant.

## 4 Dew – Constant-Sized PCS with Logarithmic Verifier

We prove our main result on PCS in this section. To go from IPC in Sect. 3 to a PCS of constant size and logarithmic verification time, we need two main ideas. First, we use Kronecker products test vectors to improve verification time from linear to logarithmic. But this breaks extractability of the new test. To recover extractability, we prove a new extremal combinatorial bound that enables us to prove a structure theorem despite the exponentially smaller randomness in the verifier’s test vectors.

### 4.1 Dew: Our Polynomial Commitment Scheme

To construct Dew, we use ideas based on Kronecker products to define new query vectors in the **TEST** and **IPP** protocols from Sect. 3 and call the modified protocols **logTEST** and **logIPP**. These changes are to bring the verifier complexity down to logarithmic in the degree of the polynomial. For notational simplicity, let the degree of the polynomial  $d = l - 1$ . We change the blue messages in the **TEST**, **IPP** protocols as below. In the **logTEST** protocol, the query vector  $\mathbf{z}$  in Fig. 2 is now redefined using just  $2 \log l$  random elements in  $\mathbb{Z}_p$ :

1. Sample random  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{\log l}$  from  $\mathbb{Z}_p^2$  where  $\mathbf{x}_j = (x_{j,0}, x_{j,1})$ .
2. For  $0 \leq k \leq l - 1$ , let  $(k_0, \dots, k_{\log l - 1})$  be the base-2 representation of  $k$  so that  $k = k_0 \cdot 2^0 + \dots + k_{\log l - 1} \cdot 2^{\log l - 1}$ . Then,

$$z_k \equiv z_{k_0, \dots, k_{\log l - 1}} := \prod_{j=1}^{\log l} x_{j, k_{j-1}}. \tag{10}$$

For **logIPP**, the query vector  $\mathbf{q}$  in Fig. 3 defined by the evaluation point  $x \in \mathbb{Z}_p$  is modified as follows

$$q_k := \prod_{0 \leq j \leq \log l - 1} (x^{k_j 2^j} \bmod p). \tag{11}$$

Note that  $0 < z_k, q_k < p^{\log l}$  and  $q_k \bmod p = x^k \bmod p$  for all  $k$ .

Our PCS Dew = (setup, commit, open, eval) is now constructed as follows:

- **setup**( $1^\kappa, D$ ): Here,  $\kappa$  is the security parameter and  $D$  is an upper bound on the degree of the committed polynomial. Sample a group of unknown order (we use class groups)  $\mathbb{G} \leftarrow \text{GGen}(\kappa)$  and a random  $g \leftarrow_s \mathbb{G}$ . Define  $\alpha := p^{2D \log D}$  ( $p$  is a large prime such that  $\text{len}(p) = \text{poly}(\kappa)$ ).  
Return  $\text{pp} = (\kappa, \mathbb{G}, g, p)$ .
- **commit**( $\text{pp}, D, f(X) \in \mathbb{Z}_p[X], l - 1$ ): Define the commitment  $C := g^{\sum_{i=0}^{l-1} f_i \alpha^{2^i}}$ , where  $f_i$  are the coefficients of the degree  $(l - 1)$  polynomial  $f(X)$  considered as integers from  $[0, p - 1]$  and the sum in  $\mathbb{Z}$ . If  $l - 1 \leq D$ , return  $(C, \mathbf{f})$ , else return error.
- **open**( $\text{pp}, D, f(X) \in \mathbb{Z}_p[X], l - 1, C, \tilde{\mathbf{f}}$ ): Check that
  - (i)  $l - 1 \leq D$ ,  $\tilde{f}_i = f_i \bmod p$  where  $f_i \in \mathbb{Z}_p$  are the coefficients of  $f(X)$ .
  - (ii)  $C = g^{\sum_{i=0}^{l-1} \tilde{f}_i \alpha^{2^i}}$ ,  $\sum_{i=0}^{l-1} \tilde{f}_i \alpha^{2^i} \in \mathbb{Z}$ , and  $\tilde{\mathbf{f}} \in \mathbb{Q}(\log l + 1, l + 1)^l$ .

Recall that

$$\mathbb{Q}(\beta_1, \beta_2) := \left\{ \frac{a}{b} : \text{gcd}(a, b) = \text{gcd}(b, p) = 1, 0 < b < p^{\beta_1}, |a/b| \leq \beta_2 \alpha \right\},$$

where  $|a|$  denotes the absolute value of the integer  $a$ .

**return 1** if all checks (i)-(iii) above pass, else **return 0**.

- **eval**( $\text{pp}, D, C, l - 1, x, v; f(X)$ ): The eval protocol consists of two sub-protocols **logTEST** and **logIPP**:
  - If  $l - 1 > D$ , **return 0**
  - Else Run **logTEST**( $C, l - 1; f(X)$ ) and **logIPP**( $C, l - 1, x, v; f(X)$ ).  
**return 1** if both these protocols accept else **return 0**.

The protocols **logTEST** and **logIPP** are simply variants of **TEST** and **IPP** in Figs. 2 and 3 by replacing the (blue messages) query vectors with Kronecker products of shorter vectors as in (10) and (11). We also replace all expensive group exponentiations for the verifier by invocations of Wesolowski’s PoE protocol. The full protocols thus obtained are presented as figures in the Appendix of [1]. In the NI version, the random query vector is derived from the RO instead of being sent in **logTEST**.

**Non-interactive Dew Using Fiat-Shamir.** Note that even though **logTEST** and **logIPP** are described as separate protocols for ease of exposition, they are both run as part of **eval**. Protocol **eval** is public-coin and we can use the Fiat-Shamir heuristic [16] to obtain a non-interactive version in the ROM that has constant-sized proof and logarithmic verification. The prover applies the RO on the commitment  $C$  to obtain  $\mathbf{x}_1, \dots, \mathbf{x}_{\log l}$ , and the random query vector  $\mathbf{z}$  is computed as described in Eq (10). The non-interactive (NI) transcript



consists of all the elements communicated in both protocols along with the NI versions of **PoE** and **PoKPE** from both protocols. Hence, the transcript communicated is  $\pi = ((C, A, \Lambda, \Gamma, R)_{\mathbf{logTEST}}, (B, N, A, \Gamma, R, S)_{\mathbf{logIPP}}, \pi_{\mathbf{PoE}}, \pi_{\mathbf{PoKPE}})$  where  $\pi_{\mathbf{PoE}}$  consists of NI transcripts of steps (4, 20, 21) and (2, 16, 17, 18) in **logTEST** and **logIPP** respectively, and  $\pi_{\mathbf{PoKPE}}$  consists of NI transcripts of steps (13, 14, 15, 16, 17) and (10, 11, 12, 13) in **logTEST** and **logIPP** respectively (from the figures in the appendix of [1]). It is easy to see that the Fiat-Shamir transformed NI transcript is succinct since the vectors  $\mathbf{x}_1, \dots, \mathbf{x}_{\mathbf{log} l}$  are now generated using the RO.

Proof of completeness is analogous (taking care of the new choices of parameters) to that of **IPP** in Theorem 31 and is deferred to the appendix of the full version [1]. Since the commitment scheme remains unchanged, the proof of binding remains as in Theorem 32. Proof of Extractability is shown in Sect. 4.2, and proof of succinctness is given in Sect. 4.3. The appendix of the full version contains concrete estimates of proof sizes. It also contains a section on how to achieve hiding and zero-knowledge evaluation for the commitment scheme.

### 4.2 Proof of Extractability of Dew

Define

$$S_{\mathbf{log}} := \left\{ \frac{m\alpha - n}{k} : m, n, k \in \mathbb{Z}, \gcd(m, k) = \gcd(k, p) = 1, 0 < m \leq k < p^{\mathbf{log} l}, \right. \\ \left. -l < n < k + l \right\}$$

as a subset of  $\mathbb{Z}$  and functions  $\chi_m, \chi_n : S_{\mathbf{log}}^q \rightarrow \mathbb{Q}^q$  which isolates the vector of fractions  $m/k$  and  $n/k$  from the elements of  $S_{\mathbf{log}}^q$ :

$$\mathbf{v} \in S_{\mathbf{log}} \implies \mathbf{v} = \left( \frac{m_i\alpha - n_i}{k_i} \right)_i, \quad \chi_m(\mathbf{v}) := \left( \frac{m_i}{k_i} \right)_i, \quad \text{and} \quad \chi_n(\mathbf{v}) := \left( \frac{n_i}{k_i} \right)_i.$$

**Theorem 41.** *The polynomial commitment scheme Dew satisfies Extractability (Def. 27) in the Generic Group Model.*

*Proof.* The proof of this theorem consists of two theorems about **logTEST** and **logIPP**. Both theorems rely on the fact that the adversary is *generic*.

**Theorem 42.** *If the Verifier in logTEST outputs accept with non-negligible probability over the choice of random  $\mathbf{z}_1, \dots, \mathbf{z}_{\mathbf{log} l} \in \mathbb{Z}_p^2$ , there exists an efficient extractor that outputs  $\mathbf{c}, \mathbf{d} \in [\alpha]^l$  such that  $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$  and  $\mathbf{d} \in S_{\mathbf{log}}^l$ .*

**Theorem 43.** *If the Verifier in logIPP outputs accept with non-negligible probability and the Verifier of logTEST also did so, there exists an extractor that outputs an opening  $\tilde{f} \in \mathbb{Z}_p[x]$  and an opening hint  $\tilde{\mathbf{c}}$  in  $\mathbb{Q}(\mathbf{log} l + 1, l + 1)$  for  $C$  such that  $v = \tilde{f}(x)$ .*

The proof of Theorem 42 is presented in Sect. 4.2.2. The proof of Theorem 43 is almost identical to that of Theorem 35 and we defer it to the full version. There are only two changes:  $d_i \in S_{\log}$  instead of  $S$  implies that the extracted vector  $\in \mathbb{Q}(\log l + 1, l + 1)^l$  instead of  $\mathbb{Q}(2, 3)^l$ , and the bounds on  $N, N', M'$  are different, leading to a lower bound  $\alpha > p^{2l(\log l)}$ .

### 4.2.1 d-Cancellation Structures

Before we present the proof of Theorem 42 in Sect. 4.2.2, we define certain combinatorial structures and state an extremal bound about them that plays a crucial role in our extractability proof. These are **d-cancellation structures** and are generalizations of  $d$ -dimensional hyper-rectangles /boxes. For instance, a 2-cancellation structure is a parallelogram, while a 3-cancellation structure can be seen as two parallel parallelograms with the same base length and height (note that this is more general than a parallelepiped - in a parallelepiped, the two parallelograms have to be congruent). For general  $d$ , we have the following recursive definition.

**Definition 44 (d-cancellation structure).** *Given a  $d$ -tuple  $(a_1, \dots, a_d) \in [n]^d$ , a **d-cancellation structure** is defined to be the set of  $2^d$  points mapped to the leaves of a depth- $d$  binary tree, where the mapping from  $[n]^d$  to nodes of the tree is recursively defined as follows.*

- Map  $(a_1, \dots, a_d)$  to the root (depth 0).
- Suppose  $(b_1, \dots, b_d)$  is mapped to a node  $u$  at depth  $d - j + 1$ . Then, for some  $y_{u,j} \in [n]$ , map  $(b_1, \dots, b_{j-1}, y_{u,j} + b_j, \dots, b_d)$  to  $u$ 's left child and  $(b_1, \dots, b_{j-1}, y_{u,j}, \dots, b_d)$  to  $u$ 's right child.

Informally, when we start from the same  $d$ -tuple, we get “similar” **d-cancellation structures** (which form an equivalence class; see an equivalent definition in the full version [1]). This is useful in counting arguments about them such as the one below.

Our main result on **d-cancellation structures** states that in the  $[n]^d$  integer lattice, we can choose at most  $n^d - (n - 1)^d \leq dn^{d-1}$  points that do not contain a **d-cancellation structure**.

**Theorem 45.** *The maximum number of points in  $[n]^d$  such that no subset of them forms a **d-cancellation structure** is  $N_d := n^d - (n - 1)^d$ . This bound is tight.*

In the extractability proof in the next section, we will argue that if the prover succeeds with non-negligible probability, then it must populate an appropriately chosen lattice with more points than this bound, leading to the existence of a **log l-cancellation structure**. We then use higher-dimensional/multilinear analogs of ideas in Theorem 34 to induce cancellations among the  $l$  equations at the points in this **log l-cancellation structure** to derive an equation with a single **d** coordinate, thereby deducing the required structure for it. Specifically, we traverse the corresponding tree bottom-up (from leaves to root) by “folding” equations from one level to the next – subtract them pairwise to reduce their number by half and eliminate half of the remaining terms in each of them. Details of this process appear in the next section and in the full version [1].

**4.2.2 Proof of Theorem 42 – logTEST Extraction**

Similar to the process in Lemma 3, we define the extractor  $\text{Ext}_T$  to first invoke the PoKPE extractor for  $C$ , which outputs  $c > 0$  such that  $C = g^c$ . Since  $E$  also passes the PoKPE protocol and  $C \cdot E = g^{\alpha^{2l-1}}$ , we infer that  $c < \alpha^{2l-1}$ . Consider the base- $\alpha$  representation of  $c$ , which is a  $2l$ -length vector.  $\text{Ext}_T$  then outputs the even indexed coordinates as  $\mathbf{c}$  and the odd indexed coordinates as  $\mathbf{d}$ .

Note that by definition, the first condition in the theorem is satisfied:  $C = g^{\sum_{i=0}^{l-1} (c_i + \alpha d_i) \alpha^{2i}}$ . An honest prover would clearly choose  $d_i = 0$  and  $c_i = x_i$ ,  $0 \leq c_i \leq p - 1$  for  $i \in [l]$  to commit to a vector  $\mathbf{x} \in \mathbb{Z}_p^l$ . However, with a cheating prover, we are only guaranteed (at this point) that  $0 \leq c_i, d_i \leq \alpha - 1$ .

Now we use the checks done by the **logTEST** verifier to derive conditions on the above extracted vector and show the second part of the theorem –  $\mathbf{d} \in S_{\log}^l$ . Suppose the prover succeeds with a non-negligible probability over the random choice of  $\mathbf{z}_1, \dots, \mathbf{z}_{\log l}$  from  $\mathbb{Z}_p^2$ .

Fix an arbitrary index  $0 \leq k \leq l - 1$ , equivalently its binary representation  $(k_1, \dots, k_{\log l})$ . Consider the partition of the space  $\mathbb{Z}_p^{2 \log l}$  by sets of the form

$$T_{\mathbf{q}} := \{(x_{1,k_1}, \dots, x_{\log l, k_{\log l}}, \mathbf{q}) : x_{j,k_j} \in \mathbb{Z}_p, 1 \leq j \leq \log l\} \text{ for } \mathbf{q} \in \mathbb{Z}_p^{\log l}.$$

Since the success probability of the prover is non-negligible, it is at least  $\frac{\log l}{p}$ . Hence, at least one of these sets (which are  $\log l$ -dimensional spaces) must have more than  $\log lp^{\log l-1} \geq p^{\log l} - (p - 1)^{\log l}$  accepting points, which implies by Theorem 45 that there exists a **log l-cancellation structure** in this space consisting of  $l$  accepting points.

For some fixed  $1 < a_j < p$  and for all  $g_1 g_2 \dots g_{\log l} \in \{0, 1\}^{\log l}$ , let this **log l-cancellation structure** be represented by

$$B := \left\{ (y_{1,g_1,g_2,\dots,g_{\log l-1}} + g_{\log l} a_1, \dots, y_{j,g_1,g_2,\dots,g_{\log l-j}} + g_{\log l-j+1} a_j, \dots, y_{\log l} + g_1 a_{\log l}) : 1 \leq j \leq \log l, y_{\dots} \in \mathbb{Z}_p \right\}$$

All the points in  $B$  can be considered as the leaves of a binary tree, with leaves indexed as  $g_1 g_2 \dots g_{\log l}$ . Starting from the root, at each node, the left child is labeled 1 and the right child is labeled 0. Thus the leftmost leaf would have index  $11 \dots 1$ , and the rightmost leaf will have index  $00 \dots 0$ .

Now, Lemma 3 gives us equations corresponding to each accepting point on the **log l-cancellation structure** relating  $\mathbf{c}, \mathbf{d}$  (given by  $\text{Ext}_T$ ) and the random variables  $x_{j,i_j}$ . We recall the definition of the query vector  $\mathbf{z}$  from Eq. (10), where for each coordinate  $z_i$  if the binary representation of  $i = i_1 \dots i_{\log l}$ , then

$$z_i \equiv z_{i_1, i_2, \dots, i_{\log l}} := \prod_{j=1}^{\log l} x_{j, i_j}$$

$$\langle \mathbf{d}, \mathbf{z} \rangle \pmod{\alpha} + \left\lfloor \frac{\langle \mathbf{c}, \mathbf{z} \rangle}{\alpha} \right\rfloor + u_1 \pmod{\alpha} = 0 \pmod{\alpha}$$

The term  $\langle \mathbf{d}, \mathbf{z} \rangle \bmod \alpha$  can be expanded as follows:

$$\begin{aligned} \langle \mathbf{d}, \mathbf{z} \rangle \bmod \alpha &= \sum_{i_1, i_2, \dots, i_{\log l} \in \{0,1\}} d_{i_1, i_2, \dots, i_{\log l}} \cdot \prod_{j=1}^{\log l} x_{j, i_j} \bmod \alpha \\ &= \sum_{i_1, \dots, i_{\log l} \in \{0,1\}} d_{i_1, \dots, i_{\log l}} \cdot \prod_{j=1}^{\log l} (y_{j, g_1, \dots, g_{\log l-j}} + g_{\log l-j+1} a_j) \cdot \prod_{\substack{j=1 \\ i_j \neq k_j}}^{\log l} x_{j, i_j} \bmod \alpha \end{aligned}$$

This expansion holds at height  $\log l$  (for all leaves  $g_1 g_2 \dots g_{\log l}$ ). To obtain the required conditions on  $\mathbf{d}$ , we subtract the  $l$  equations in a specific order to cancel out all but one term. This is possible due to the fact that the coefficients of  $d_{i_1 \dots i_{\log l}}$  are multilinear in each of the randomly sampled variables.

More precisely, at any intermediate height in the binary tree, we obtain the equation at that node by subtracting the equation at the right child from the equation at the left child. For instance, at height  $(\log l - 1)$ , the first term takes the form :

$$a_1 \sum_{i_1, \dots, i_{\log l-1}} d_{k_1, i_2, \dots, i_{\log l}} \cdot \prod_{j=2}^{\log l} (y_{j, g_1, \dots, g_{\log l-j}} + g_{\log l-j+1} a_j) \cdot \prod_{\substack{j=2 \\ i_j \neq k_j}}^{\log l} x_{j, i_j} \bmod \alpha$$

In general, we get at height  $0 \leq t < \log l$ ,

$$\begin{aligned} &\prod_{j=1}^{\log l-t} a_i \sum_{i_1, i_2, \dots, i_t \in \{0,1\}} d_{k_1, k_2, \dots, k_{\log l-t}, i_{\log l-t+1}, \dots, i_{\log l}} \\ &\cdot \prod_{j=\log l-t+1}^{\log l} (y_{j, g_1, g_2, \dots, g_{\log l-j}} + g_{\log l-j+1} a_j) \cdot \prod_{\substack{j=\log l-t+1 \\ i_j \neq k_j}}^{\log l} x_{j, i_j} \bmod \alpha \end{aligned}$$

Notice that at the root, i.e., at height 0, we are left with the single term  $a_1 \dots a_{\log l} \cdot d_{k_1, k_2, \dots, k_{\log l}}$ .

For the rest of the folded equation, we only use bounds on the other terms and not the exact expression. The actual expression is a symbolic subtraction of the floor terms and the ‘ $u$ ’ terms. This is similar to what is done in Theorem 34 generalised to higher dimensions.

Specifically, indexing the  $l$  points/leaves by  $\mathbf{z}_{g_1 g_2 \dots g_{\log l}}$ , we get the expression for the remaining two terms (call this expression  $n$ ) as

$$\left( \sum_{g_1, g_2, \dots, g_{\log l} \in \{0,1\}} (-1)^{\sum_{j=1}^{\log l} i_j} \cdot \left[ \frac{\langle \mathbf{C}, \mathbf{z}_{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_{\log l}} \rangle}{\alpha} \right] + \sum_{g_1, g_2, \dots, g_{\log l} \in \{0,1\}} (-1)^{\sum_{j=1}^{\log l} i_j} \cdot u_{g_1, g_2, \dots, g_{\log l}} \right) \bmod \alpha$$

Since for all  $x$ ,  $x - 1 \leq \lfloor x \rfloor < x$  and  $u \in \{0, 1\}$ , we can bound the above expression  $n$  by

$$\frac{c_{k_1, \dots, k_{\log l}} \cdot \prod_{i=1}^{\log l} a_i}{\alpha} - 2^{\log l} < n < \frac{c_{k_1, \dots, k_{\log l}} \cdot \prod_{i=1}^{\log l} a_i}{\alpha} + 2^{\log l}$$

$$\implies -l < n < \prod_{i=1}^{\log l} a_i + l$$

Hence, there exists  $m$  such that  $d_{k_1, \dots, k_{\log l}} = \frac{m\alpha - n}{a_1 \dots a_{\log l}}$  where  $m \leq \prod_{i=1}^{\log l} a_i < p^{\log l}$  (as  $d_{k_1, \dots, k_{\log l}} < \alpha$ ) and  $-l < n < \prod_{i=1}^{\log l} a_i + l$  as shown above.

Hence,  $d_{k_1, \dots, k_{\log l}} \in S_{\log}$ . Since  $(k_1, \dots, k_{\log l})$  was arbitrary,  $\mathbf{d} \in S_{\log}^l$ .

### 4.3 Succinctness of Dew

**Theorem 46 (Proof and Verifier Succinctness).** *In Dew, the commitment and evaluation proof sizes are  $\text{poly}(\kappa)$  and the Verifier runs in time  $\text{poly}(\kappa) \cdot \log(l)$ .*

*Proof.* Proof succinctness is easy to see; the commitment is a single group element and the evaluation protocol only communicates a constant number of group elements (the PoE protocols are also constant-sized) and the query vector elements  $\mathbf{x}_1, \dots, \mathbf{x}_{\log l}$ . However, as mentioned before, in the NI version, the  $2 \log l$  random field elements are generated using a RO – this makes the proof size of the non-interactive version of the protocol constant.

To analyse the verifier computation, notice that the the only potentially expensive computations are the computation of  $\sigma$  and raising group elements to large powers (the PoKPE protocols consist of a constant number of PoKE protocols, which are efficient). The group exponentiations are made more efficient for the verifier by engaging in a constant number of PoE protocols with the prover. The only remaining bottleneck is the computation of  $\sigma \pmod q$  for some prime  $q$  in the invocation of Weslowski’s PoE (for  $\sigma$  in both **logTEST** and **logIPP**).

In **logTEST**, using the definition of the test vector in (10) and direct manipulation implies that  $\sigma \pmod q$  can be computed in  $O(\log l)$  time as follows

$$\sigma \pmod q = \sum_{k=0}^{l-1} \alpha^{2^{l-2-2k}} z_k \pmod q = \alpha^{2^{l-2}} \prod_{i=1}^{\log l} \left( x_{i,0} + x_{i,1} \alpha^{-2^{i+1}} \right) \pmod q$$

In **logIPP**, by a similar manipulation as above using the definition of the the query vector in (11), we obtain

$$\sigma = \sum_{k=0}^{l-1} \alpha^{2^{l-1-2k}} q_k \pmod q = \alpha^{2^{l-1}} \prod_{i=0}^{\log l-1} \left( 1 + (x^{2^i} \pmod p) \alpha^{-2^{i+1}} \right) \pmod q$$

Also note that for efficient computation, we need to compute  $\alpha \pmod q$  and  $\alpha^{-1} \pmod q$  (If  $\alpha^{-1} \pmod q$  does not exist, then  $\alpha = 0 \pmod q$  and computing

$\sigma$  becomes trivial). In this case, since  $\alpha = p^L$  for some  $L = O(l)$ , computing  $\alpha \bmod q = p^L \bmod q$  can be efficiently done in  $O(\log l)$  time using repeated squaring. Once this is found,  $\alpha^{-1} \bmod q$  can also be efficiently found using the Extended Euclidean algorithm.

## 5 Transparent zkSNARKs via Dew

As a corollary of our PCS, we get concrete instantiations of new *transparent succinct arguments* by compiling an information theoretic proof in an idealized model into a succinct argument using a PCS.

The modular approach advocated for designing efficient arguments consists of two steps; constructing an information theoretic protocol in an abstract model (PCP, linear PCP, IOP etc.), and then compiling the information-theoretic protocol via a cryptographic compiler to obtain an argument system. Many recent constructions of zkSNARKs [9, 12, 17] follow this approach where the information theoretic object is an algebraic variant of IOP, and the cryptographic primitive in the compiler is a polynomial commitment scheme. Marlin [12] uses an IOP abstraction called algebraic holographic proofs (AHP), and [9] uses an abstraction called polynomial IOPs (PIOPs). In both these abstractions, the prover and the verifier interact where the prover provides oracle access to a set of polynomials, and the verifier sends random challenges. Then, the verifier asks for evaluations of these polynomials at these challenge points and decides to accept or reject based on the answers. PLONK [17] uses an abstraction called idealized low degree protocols (ILDPs) that proceeds in a similar way except that at the end of the protocol, the verifier checks a set of polynomial identities over the oracles sent by the prover. Polynomial Holographic IOP (PHP) [11] specializes the IOP notion in two ways (i) it is holographic – that is, the verifier has access to a set of oracle polynomials created during the setup phase that encode the relation, (ii) the verifier can directly check polynomial identities. The high level idea to build a zkSNARK with universal SRS starting from PIOPs/AHPs/ILDPs/PHPs is the following: the argument prover commits to the polynomials obtained from the information-theoretic prover, and then uses the evaluation opening property of the polynomial commitment scheme to respond to the evaluation queries of the verifier in a verifiable way.

We present concrete instantiations of zkSNARKs obtained by using our transparent PCS to cryptographically compile the AHP underlying the constructions of Sonic, Marlin and PLONK. We present the details and compare our zkSNARK Dew-SNARK to existing schemes in the full version [1].

## References

1. Arun, A., Ganesh, C., Lokam, S., Mopuri, T., Sridhar, S.: Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Report 2022/419 (2022). <https://eprint.iacr.org/2022/419>

2. Ben-Sasson, E., Chiesa, A., Goldberg, L., Gur, T., Riabzev, M., Spooner, N.: Linear-size constant-query iops for delegating computation. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11892, pp. 494–521. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-36033-7\\_19](https://doi.org/10.1007/978-3-030-36033-7_19)
3. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Time- and space-efficient arguments from groups of unknown order. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 123–152. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84259-8\\_5](https://doi.org/10.1007/978-3-030-84259-8_5)
4. Bollobás, B.: Extremal Graph Theory. Reprint of the 1978 original. Dover Publications, Inc., Mineola, NY (2004) ISBN: 0-486-43596-2
5. 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. LNCS, vol. 11692, pp. 561–586. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26948-7\\_20](https://doi.org/10.1007/978-3-030-26948-7_20)
6. Bosma, W., Stevenhagen, P.: On the computation of quadratic 2-class groups. *Journal de Théorie des Nombres de Bordeaux* **8**(2), 283–313 (1996)
7. Buchmann, J., Hamdy, S.: A survey on iq cryptography. In: Proceedings of Public Key Cryptography and Computational Number Theory, pp. 1–15 (2001)
8. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. Cryptology ePrint Archive, Report 2019/1229 (2019). <https://eprint.iacr.org/2019/1229>
9. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_24](https://doi.org/10.1007/978-3-030-45721-1_24)
10. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45708-9\\_5](https://doi.org/10.1007/3-540-45708-9_5)
11. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: a toolbox for more efficient universal and Updatable zkSNARKs and commit-and-prove extensions. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13092, pp. 3–33. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92078-4\\_1](https://doi.org/10.1007/978-3-030-92078-4_1)
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. LNCS, vol. 12105, pp. 738–768. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_26](https://doi.org/10.1007/978-3-030-45721-1_26)
13. Chu, H., Fiore, D., Kolonelos, D., Schröder, D.: Inner product functional commitments with constant-size public parameters and openings. In: Galdi, C., Jarecki, S., (eds.) Security and Cryptography for Networks, pp. 639–662. Springer International Publishing, Cham (2022). [https://doi.org/10.1007/978-3-031-14791-3\\_28](https://doi.org/10.1007/978-3-031-14791-3_28)
14. 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](https://doi.org/10.1007/3-540-46035-7_17)
15. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), pp. 427–438. IEEE (1987)
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](https://doi.org/10.1007/3-540-47721-7_12)

17. 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://ia.cr/2019/953>
18. 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](https://doi.org/10.1007/978-3-642-17373-8_11)
19. Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 530–560. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26948-7\\_19](https://doi.org/10.1007/978-3-030-26948-7_19)
20. Lee, J.: Dory: efficient, transparent arguments for generalised inner products and polynomial commitments. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13043, pp. 1–34. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-90453-1\\_1](https://doi.org/10.1007/978-3-030-90453-1_1)
21. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D., (eds.) ICALP 2016, vol. 55 of LIPIcs, pp. 30:1–30:14. Schloss Dagstuhl, (July 2016)
22. Micali, S., Rabin, M., Kilian, J.: Zero-knowledge sets. In: 44th Annual IEEE Symposium on Foundations of Computer Science 2003, Proceedings, pp. 80–91. IEEE (2003)
23. Ràfols, C., Zapico, A.: An algebraic framework for universal and updatable SNARKs. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12825, pp. 774–804. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84242-0\\_27](https://doi.org/10.1007/978-3-030-84242-0_27)
24. Rosenfeld, L.: The box problem in two and higher dimensions. Bachelor’s thesis, University of Rochester (2016). <https://www.sas.rochester.edu/mth/undergraduate/honorspaperspdfs/rosenfeldhonorsthesis16.pdf>
25. 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)
26. Wesolowski, B.: Efficient verifiable delay functions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 379–407. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17659-4\\_13](https://doi.org/10.1007/978-3-030-17659-4_13)
27. 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)



**IO and ZK II**



# Non-Interactive Publicly-Verifiable Delegation of Committed Programs

Riddhi Ghosal<sup>1</sup>(✉), Amit Sahai<sup>1</sup>, and Brent Waters<sup>2,3</sup>

<sup>1</sup> UCLA, Los Angeles, USA  
{riddhi,sahai}@cs.ucla.edu

<sup>2</sup> UT Austin, Austin, USA  
bwaters@cs.utexas.edu

<sup>3</sup> NTT Research, Palo Alto, USA

**Abstract.** In this work, we present the first construction of a fully non-interactive publicly-verifiable delegation scheme for committed programs. More specifically, we consider a setting where Alice is a trusted author who delegates to an untrusted worker the task of hosting a program  $P$ , represented as a Boolean circuit. Alice also commits to a succinct value based on  $P$ . Any arbitrary user/verifier *without knowledge of*  $P$  should be convinced that they are receiving from the worker an actual computation of Alice's program on a given input  $x$ .

Before our work, the only object known to imply this challenging form of delegation was a SNARG/SNARK for  $\mathcal{NP}$ . This is because from the point of view of the user/verifier, the program  $P$  is an unknown witness to the computation. However, constructing a SNARG for  $\mathcal{NP}$  from standard assumptions remains a major open problem.

In our work, we show how to achieve delegation in this challenging context assuming only the hardness of the Learning With Errors (LWE) assumption, bypassing the apparent need for a SNARG for  $\mathcal{NP}$ .

## 1 Introduction

We consider a scenario where a trusted software author Alice wishes to make it possible for a set of users to make use of her program  $P$ , which we treat as a (non-uniform) Boolean circuit. In particular, this program  $P$  may have embedded within it a large proprietary database that Alice's program makes use of. However, Alice neither wants to release her program  $P$  nor does she want to host and execute the program herself. Instead she wishes to delegate this computation to an untrusted Worker, and the User/Verifier wants to be certain that they are receiving an output obtained via a computation of Alice's actual program  $P$ . As illustrated in Fig. 1, the way this works is:

1. Alice sends the program  $P$  along with some computed **state** to the Worker, and Alice also publishes a succinct hash  $H_P$  of her program, which the User/Verifier obtains. This step is done once and for all.
2. An Input Provider chooses an input  $x$ , which is sent to both the Worker and the User/Verifier. Note that the input provider could be some public source

of information like a news channel or bulletin board, and need not involve the User/Verifier.

3. Finally, the Worker computes the output  $y = P(x)$  along with a succinct proof  $\Pi$ , and sends both of these to the User/Verifier. Steps 2 and 3 may be repeated polynomially many times.

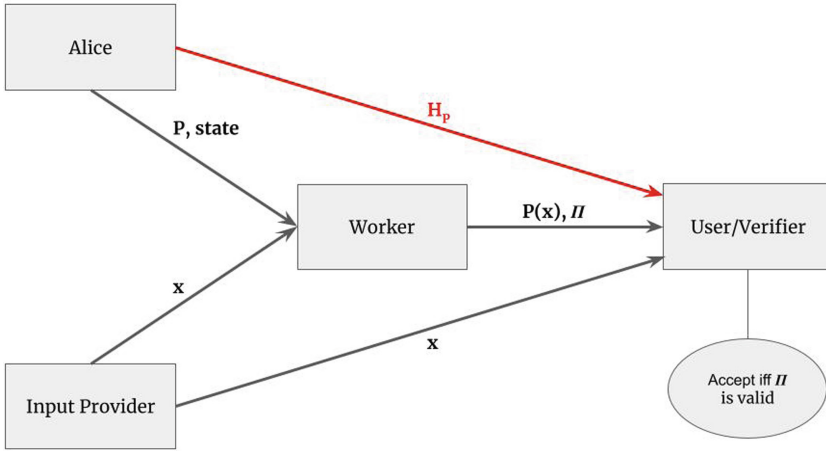


Fig. 1. The Delegation Setup

As illustrated in Fig. 1, this process involves no back-and-forth communication. The communication is entirely unidirectional – which we call non-interactive – from left to right. Furthermore, we say that this scenario is *succinct* if all communication to the User/Verifier, and the runtime of the User/Verifier, is  $\text{poly}(\log |P|, \lambda, |x|)$ , where  $\lambda$  is a security parameter.

*Remark 1.* Note that on one hand, the Worker is trusted with the program  $P$  by Alice, whereas, it is not trusted by the verifier. This asymmetry of trust is inherent in our setup and is well motivated. In a typical real world situation, the verifier is typically a user on the internet who takes part in a one off interaction with a cloud service for some computation. The need to prove honesty in this situation is significant. On the other hand, Alice might be able to have an agreement with the cloud service before handing over her program, which would make it hard for their Worker to breach trust without consequences.

*Comparison to Prior Work.* What we have just described is one of the most challenging variants of the classical problem of *publicly verifiable delegation* which has been the subject of intense work for decades, for many relaxed variations of the model that we describe above.

Specifically, delegation schemes *without public verification* based on standard assumptions for deterministic and non-deterministic computations have been

designed [1, 6, 7, 11, 12, 22, 24, 25, 37–39]. Restricting verification to a *designated* verifier implies that the worker needs to produce a fresh proof unique for each particular verifier for any computation, which is certainly not ideal. Another line of work [15] achieves public verification but does not achieve public delegation. In other words, the input provider needs to run a pre-processing algorithm corresponding to the program  $P$  before being able to delegate. Another model which has been extensively explored is when the User/Verifier is allowed to have interaction with the Worker, i.e., *interactive delegation*. Influenced by the first work on interactive efficient arguments by Kilian [27], there have been several works from standard assumptions [5, 24, 33, 34] and some even unconditional soundness [17, 36]. These are however not applicable in our setting where only one-way communication is permitted between the parties, as can be seen in the acyclic graph in Fig. 1.

With regard to *non-interactive publicly verifiable delegation*, Starting from the seminal work on computationally sound proofs by Micali [31] in the random oracle model, there have been several constructions on publicly verifiable non-interactive delegation schemes [2–4, 13, 16, 18, 28, 32] based on the *Random Oracle Model* or non-standard *knowledge assumptions*. From more standard assumptions, there have been several works recently [1, 6, 7, 23]. An illustrative example is the recent work of [23] that proposed the first publicly verifiable non-interactive delegation scheme from a falsifiable decisional assumption on groups with bilinear pairings. However, in contrast with the setting we describe above, they can only achieve succinct delegation when the Verifier *knows the program  $P$* . In our setting of Boolean circuits, this trivializes the delegation problem, since reading  $P$ 's description takes as long as evaluating  $P$ . Indeed, the case that we consider—where Alice's program is large—is extremely well motivated: the program  $P$  could be an ML model with billions of painstakingly learned parameters.

*The SNARGs for  $\mathcal{NP}$  barrier.* Why has constructing a protocol that caters to the fully non-interactive setting which we have defined been so elusive? Note that in our problem, the User/Verifier and Input Provider do not know the program  $P$ . Hence, from User/Verifier's perspective,  $P$  is an  $\mathcal{NP}$  witness. Thus, it certainly seems that finding a solution is intricately related to a major goal in the area of non interactive succinct proof systems, i.e., *SNARGs for  $\mathcal{NP}$* . Unfortunately, the only known constructions of SNARGs for  $\mathcal{NP}$  base their soundness on the *Random Oracle Model* or non-standard *knowledge assumptions*. Finding a solution solely relying on standard assumptions has been an open problem for over a decade. In fact, the closest that we have come is the very recent work achieving SNARGs for  $\mathcal{P}$  [10] (see also [26]).

The major technical contribution in our work is to enable *Non-Interactive Publicly Verifiable Succinct Delegation for Committed Programs* without having to use SNARGs for  $\mathcal{NP}$ .

*Our Contribution:* We present the first complete solution to achieving succinct non interactive publicly verifiable delegation for committed programs. Indeed, furthermore, we can also achieve zero-knowledge guarantees as well. Our only computational assumption is the hardness of the *Learning with Errors* (LWE)

problem. Somewhat surprisingly, we show that SNARGs for  $\mathcal{NP}$  are not required to solve this problem, even though the statement being proved looks like an  $\mathcal{NP}$  statement to the Verifier!

Instead, we show that many ideas from SNARGs for  $\mathcal{P}$  [10] can in fact be applied here. Although  $P$  is unknown to the User/Verifier, we show that it suffices for Alice to communicate a tiny amount of information of size  $\text{poly}(\log |P|)$  about the program  $P$  (referred to as  $H_P$ ) as shown in Fig. 1. Because Alice is the author of  $P$ , this  $H_P$  can be *trusted* as correctly generated. We stress that Alice does not need to know  $x$  to compute  $H_P$ , hence this achieves public delegation and public verification in the completely non-interactive model described above. This leads to our main theorem,

**Theorem 1.** *Assuming the hardness of the LWE problem, Fig. 2 gives a construction for publicly verifiable non-interactive succinct delegation for committed programs with CRS size, proof size and verifier time  $\text{poly}(\lambda, \log |P|, |x|)$  and prover run time being  $\text{poly}(\lambda, |P|)$ .*

Finally, in order to get zero-knowledge, it suffices for Alice to commit to  $H_P$  rather than sending it out in the open. We then present a generic transformation to convert any delegation protocol of this form to attain zero-knowledge.

**Theorem 2.** *Assuming the hardness of the LWE problem and existence of a succinct delegation scheme, Fig. 5 gives a construction for publicly verifiable succinct delegation scheme with zero knowledge such that CRS size, proof size and verifier time are  $\text{poly}(\lambda, \log |P|)$  and prover run time is  $\text{poly}(\lambda, |P|)$ .*

Finally, we also show how to achieve *zero knowledge* versions of our delegation scheme, meeting the same strong succinctness and efficiency goals, and under the same assumption (LWE).

We present a more detailed explanation in the Technical Overview.

## 2 Technical Overview

*Our Delegation Scenario.* Let us briefly recall the setup of our delegation scenario. There are 4 parties, namely, (1) Alice-the program author  $\text{ProgAuth}$  who sends a program  $P$  and some computed state  $\text{state}$  to a Worker, (2) an Input Provider  $I$  that outputs some value  $x$ , (3) Worker  $W$  that takes as input  $(P, \text{state}, x)$  and outputs  $P(x)$  and a proof  $\Pi$ , and (4) User/Verifier  $V$  gets as inputs  $(x, P(x), \Pi)$  and outputs 1 if and only if  $\Pi$  was a valid proof. Assume that all the parties get the security parameter  $\lambda$  as an input. An additional requirement is that  $|\Pi|$  and runtime of  $V$  is  $\text{poly}(\lambda, \log |P|, |x|)$ , and  $W$  runs in time  $\text{poly}(\lambda, |x|, |P|)$ . Thus, any non-interactive publicly verifiable succinct delegation scheme can be viewed as a collection of 4 algorithms:  $\text{sDel} = (\text{ProgAuth}, W, I, V)$  with the input output behaviour and efficiency guarantees as specified. Note that this is indeed a  $\mathcal{P}$  computation for the Worker but the primary challenge is that the verifier does not have knowledge of the “witness”  $P$ , hence this is an  $\mathcal{NP}$

computation from the verifier’s point of view. In this work, we observe that it is indeed feasible to achieve our delegation scenario for all circuits without having to go through SNARGs for  $\mathcal{NP}$ . Our technique is based on the recent work of Choudhuri et al. [10] on SNARGs for  $\mathcal{P}$ . We begin by giving a brief overview of their approach and elaborate the challenges of directly incorporating their methodology for our setting.

*Challenges of Implementing* [10]. Roughly, the work of [10] uses Batch Arguments for  $\mathcal{NP}$  (BARGs), which they build from LWE. BARGs allow an efficient prover to compute a non-interactive and publicly verifiable “batch proof” of many  $\mathcal{NP}$  instances, with size  $\text{poly}(|w| \log T)$  for  $T$ -many  $\mathcal{NP}$  statements with each witness of size  $|w|$ . They begin by looking at  $P$  as a Turing machine and the steps of  $P$ ’s computation are interpreted as an *Index Circuit*  $C_{\text{index}}$ . Say,  $P$  terminates in  $T$  steps. Formally, they construct a BARG for the *Index Language*  $\mathcal{L}^{\text{index}}$ , where

$$\mathcal{L}^{\text{index}} = \{(C_{\text{index}}, i) \mid \exists w_i, \text{ such that } C(i, w_i) = 1\},$$

where  $i \in [T]$  is an index. Let  $s_0, s_1, \dots, s_T$  denote the encoding of internal states of  $P$  along with its tape information, and let *Step* be its step function such that  $\text{Step}(s_{i-1}) = s_i$ . The witness for the  $i^{\text{th}}$  intermediate computation is then defined as  $w_i = (s_{i-1}, s_i)$ . The index circuit is built such that  $(C_{\text{index}}, i) \in \mathcal{L}^{\text{index}}$  essentially implies that the Turing machine step function was correctly computed on  $s_{i-1}$  to yield  $s_i$ . Note that this alone does not suffice as a proof because the BARG only confirms that  $(s_{i-1}, s_i)$  and  $(s'_i, s_{i+1})$  are valid witnesses. If  $s_{i-1}, s_i, s'_i, s_{i+1}$  are generated by the step function of the same Turing machine  $P$ , they must be consistent with each other, i.e.,  $s_i = s'_i$ . However, this is not guaranteed by a BARG.

To resolve this issue, the prover also sends a *Somewhere Extractable Hash (SE)* to the witnesses  $(s_0, \{s_{i-1}, s_i\}_{i \in [T]})$ . The extraction property of this hash allows the verifier to check if the witness of two consecutive BARG instances are indeed consistent with each other. At this stage, we would like to remind the reader of their efficiency goals where crucially, they desire proof size and verification time to be  $\text{poly}(\lambda, \log T)$ . However, note that  $|C_{\text{index}}|$  grows linearly with  $|s_i|$  and the known constructions [20] of SE hashes can only produce hashes with size  $\text{poly}(|s_i|)$ . This means that total communication and verifier run time will be at least  $\text{poly}(|s_i|)$ . This is certainly no good if the Turing machine has massive states. To overcome this final barrier, they make use of Hash Trees which compress the states  $s_i$  to a short hash  $h_i$  such that  $|h_i| = \text{poly}(\lambda)$ . Such trees [30] also have a soundness property where a Prover must produce a succinct proof  $\Pi_i$  that the hash tree was indeed implemented correctly at the  $i^{\text{th}}$  step of the Turing machine computation. Once the succinctness guarantee is ensured, the prover then produces SE hashes corresponding to  $(h_0, \Pi_0, \{h_{i-1}, \Pi_{i-1}, h_i, \Pi_i\}_{i \in [T]})$  along with the openings to these hashes. To summarise, the proof consists of two parts, (1) The BARG proof, and (2) A *somewhere extractable* hash of the witnesses. Relying on the soundness of BARG, extraction correctness property

of SE hash and soundness of the Hash Tree, a User/Verifier can check if each of these  $T$  intermediate steps are indeed the correct states for  $P$ , i.e., the computation was done honestly.

However, this approach only works if User/Verifier can confirm that the inputs used for the computation by the Worker, i.e.  $(P, x)$  are indeed the correct starting values as provided by the Program Author and Input Provider. This works fine for [10] because in their setting, the User/Verifier actually knows  $(P, x)$ . Unfortunately, this is not at all true in our scenario. Thus, the techniques of Choudhuri et al. [10] cannot be implemented directly as the soundness of the BARG proof cannot provide any guarantees if there is no way for to check that the initial inputs used by the Worker are correct.

*Our Idea.* We start with an alternate way of interpreting the computation of  $P$  on input  $x$  as the following: Consider a Circuit-Universal Turing Machine  $\mathcal{TM}$  which takes as input  $P, x, y$  and accepts  $(P, x, y)$  in  $T = \tilde{O}(|P|)$  steps if  $P(x) = y$ . We can assume without loss of generality that  $P \in \{0, 1\}^m$ ,  $x \in \{0, 1\}^n$  and  $y \in \{0, 1\}$ , where  $m, n \leq 2^\lambda$ . Keeping this in mind, we introduce the notion of *Semi-Trusted SNARGs* for  $\mathcal{NP}$ . This new kind of SNARG is one that will work for general  $\mathcal{NP}$  computations, but only with a little bit of extra help from a trusted party that knows the witness – which in our delegation scenario is Alice, who knows the witness  $P$ !

A Semi-Trusted SNARG is a tuple of algorithms:  $\text{stSNARG} = (\text{Setup}, \text{TrustHash}, \text{P}, \text{V})$ , where (1) **Setup** is a randomised algorithm that takes as input the security parameter and outputs a Common Random String (CRS). (2) a *trusted* deterministic **TrustHash** takes as input the  $(\text{CRS}, P)$  and outputs a digest  $H_P$ , (3) a deterministic prover **P** which takes as input CRS and  $(P, x, y)$ , and outputs a proof  $\Pi$ , and (4) a deterministic verifier **V** which gets  $\text{CRS}, (H_P, x, y, \Pi)$  as input and outputs 1 iff  $\Pi$  is valid. It must be that  $|\Pi|$  and run time of **V** is  $\text{poly}(\lambda, \log T)$ , and **P** runs in time  $\text{poly}(\lambda, |x|, |P|, T)$ . A simple reduction shows that in the CRS model (or alternatively in a model where Alice chooses the CRS), existence of  $\text{stSNARG}$  implies the existence of  $\text{sDel}$ . We show this formally in Lemma 11. Hence, from here onwards, our goal is to construct a *Semi-Trusted SNARG* for  $\mathcal{NP}$ .

We briefly provide an informal explanation of our construction.

Like [10], every intermediate state of the Universal Turing Machine is encoded into a succinct hash (call it  $h_0, \dots, h_T$ ) accompanied with succinct proofs  $\{\Pi_i\}_{i \in [T]}$ . The prover computes two independent copies of *Somewhere Extractable* (SE) hashes  $(c_1, c_2)$  of the encoding  $\{h_0, \{(h_1, \Pi_1), \dots, (h_T, \Pi_T)\}\}$  along with their corresponding openings. Here  $h_0 = (\text{st}_0, H_P, H_x, H_{\text{work}})$ , where  $\text{st}_0$  is that hash of  $\mathcal{TM}$ 's starting state which is publicly known,  $H_x$  denote the hash of  $x$ , and  $H_{\text{work}}$  is the hash of  $\mathcal{TM}$ 's blank work tape. The use of two independent SE hashes are pivotal for soundness which we elaborate later.

We point out that **TrustHash** computes  $H_P$  using the same hash tree which is used for hashing the Turing machine states by the Prover. This is crucial to ensure soundness of the protocol. We show in Fig. 3 that once the public hash is fixed by **TrustHash**, one can hard code  $(y, c_1, c_2, T, H_P, H_x)$  to the index circuit

$C_{\text{index}}$  for BARG. At this point, we can now follow the approach from [10].  $V$  can rely upon the binding property/collision resistance of the hash to ensure that the prover has used  $P$  and  $x$  which were provided by Alice and the input provider respectively. The main observation here is that once a trusted party fixed a hash of the program  $P$  and  $V$  is convinced that computation was commenced with the correct inputs, the soundness of BARG, extraction correctness of the SE hash and soundness of hash tree ensures that the semi-trusted SNARG construction is sound.

While our proof of soundness closely follows the blueprint of [10], we choose to present our proof in a different, and arguably simpler, way. In [10], *No-Signaling Somewhere Extractable*(NSSE) hashes are used extensively. In our proof, we choose to omit explicit use of this notion, and instead we make direct use of two independent SE hashes as mentioned above. A simple hybrid argument then gives a straightforward proof for soundness. This shows that the “anchor and step” use of SE hashes, which dates to the introduction of somewhere-binding hashes [20] in 2015, is directly sufficient for this proof of soundness.

*Zero-Knowledge.* We have only discussed soundness guarantees thus far. However, in our delegation scenario, it might also be extremely important to ensure that no information about  $P$  leaked to  $V$  during the delegation process. Hence it is important to add *zero-knowledge* guarantees to our protocol. We finally give a generic transformation to modify a semi-trusted SNARG to add zero knowledge guarantees. In order to do so we make use of a statistically binding extractable commitment scheme and a NIZK<sup>1</sup>, and roughly make the following modifications:

- We add an additional commitment to 0 in the CRS which is never used in the proof but helps in proving zero knowledge.
- The public hash output by `TrustHash` is a binding commitment  $C_P$  of  $H_P$ . It then sends  $(P, H_P)$  to the worker  $W$  only.
- The SE hashes  $c_1, c_2$  are also committed as a part of the proof and not published in the open.
- The prover wraps the BARG  $\Pi$  with a NIZK which proves that that the BARG verification circuit indeed accepts the BARG proof.
- The Verifier then checks if the NIZK proof is valid.

The binding and hiding property of the commitment, and *witness indistinguishability* of NIZK guarantees zero knowledge.

### 3 Preliminaries

We use some standard tools as building blocks to perform the Succinct Delegation.

---

<sup>1</sup> Multi-theorem NIZK from LWE is possible by combining [35] and [14]. Note that the weaker notion NIWI would also suffice to achieve zero knowledge in our setting.



- **Somewhere Extractable Hash** [9, 10, 20]:  
SE = (SE.Gen, SE.TGen, SE.Hash, SE.Open, SE.Verify, SE.Ext)
- **Non Interactive Batch Arguments (BARG) for Index Language** [10]:  
BARG = (BARG.Gen, BARG.TGen, BARG.Prove, BARG.Verify)
- **Hash Tree** [23, 30]:  
HT = HT.Gen, HT.Hash, HT.Read, HT.Write, HT.VerRead, HT.VerWrite
- **Non Interactive Zero Knowledge Argument** [8, 19, 35]:  
(NIZK = NIZK.Gen, NIZK.Prove, NIZK.V)
- **Statistically Binding Extractable Commitment** [29]:  
Com<sub>bind</sub> = (Com.Gen, Com.TGen, Com.C, Com.Ext)

We use all the primitives in a standard way as prior works. The hash tree can be constructed from any collision resistant hash function. The others are known to be instantiated from LWE. Formal definitions and properties of the primitives can be found in the Supplementary Material.

## 4 Publicly Verifiable Non Interactive Succinct Delegation

We formally define the notion of Publicly Verifiable Non Interactive Succinct Delegation (sDel) which is similar to the definition proposed in prior works [21]. Such a delegation scheme in the CRS model involves the following PPT algorithms, (1) Software/Program Author ProgAuth (3) Cloud Worker  $W$ , and (3) Verifier  $V$ . An sDel comprises of the following polynomial time algorithms:

- $\text{sDel.Setup}(1^\lambda)$ : A randomized setup algorithm which on input security parameter  $\lambda$  and outputs  $\text{crs}$ .
- $\text{sDel.ProgAuth}(1^\lambda, \text{crs})$ : A program author which takes as input  $\lambda$ , outputs a (not public) program  $P \in \{0, 1\}^m$ ,  $m \leq 2^\lambda \in \mathbb{N}$ , state and a public digest  $H_P$ .
- $\text{sDel.W}(\text{crs}, P, \text{state}, H_P, x)$ : A deterministic cloud worker which on input  $\text{crs}$ , program  $P$ , input  $x \in \{0, 1\}^n$ ,  $n \leq 2^\lambda \in \mathbb{N}$  outputs a value  $y$  and proof  $\Pi$ .
- $\text{sDel.V}(\text{crs}, x, y, H_P, \Pi)$ : A deterministic verifier which on input  $\text{crs}$ , digest  $H_P$ ,  $x$ ,  $y$ ,  $\Pi$  either accepts or rejects.

A publicly verifiable succinct delegation scheme ( $\text{sDel.Setup}$ ,  $\text{sDel.ProgAuth}$ ,  $\text{sDel.W}$ ,  $\text{sDel.V}$ ) satisfies the following properties:

- **Completeness.** For every PPT program generating algorithm  $\text{sDel.ProgAuth}$ , every  $\lambda, n, m \in \mathbb{N}$ , and for all  $x \in \{0, 1\}^n$  such that  $n, m < 2^\lambda$ , we have

$$\Pr[\text{sDel.V}(\text{crs}, x, y, H_P, \Pi) = 1 \wedge P(x) = y \mid \text{crs} \leftarrow \text{sDel.Setup}(1^\lambda), \\ ((P, \text{state}), H_P) \leftarrow \text{sDel.ProgAuth}(1^\lambda, \text{crs}), \\ (y, \Pi) \leftarrow \text{sDel.W}(\text{crs}, P, \text{state}, H_P, x)] = 1.$$

- **Efficiency.**  $\text{sDel.Setup}$  runs in time  $\text{poly}(\lambda)$ ,  $\text{sDel.W}$  runs in time  $\text{poly}(\lambda, |P|, |x|)$  and outputs a proofs of length  $\text{poly}(\lambda, \log |P|, |x|)$ , and  $\text{sDel.V}$  runs in time  $\text{poly}(\lambda, \log |P|, |x|)$ .

- **Soundness.** For every PPT adversary  $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ , every PPT program generating algorithm  $\text{sDel.ProgAuth}$ , and the tuple  $n = n(\lambda), m = m(\lambda)$ , there exists a negligible function  $\text{negl}(\lambda)$  such that for every  $\lambda \in \mathbb{N}$ ,

$$\begin{aligned} \Pr[\text{sDel.V}(\text{crs}, x, y, H_P, \Pi) = 1 \wedge P(x) \neq y \mid \text{crs} \leftarrow \text{sDel.Setup}(1^\lambda), \\ ((P, \text{state}), H_P) \leftarrow \text{sDel.ProgAuth}(1^\lambda, \text{crs}), \\ (x, \text{aux}) \leftarrow \mathcal{A}_1(1^\lambda, \text{crs}), (y, \Pi) \leftarrow \mathcal{A}_2(\text{crs}, P, \text{state}, H_P, x, \text{aux})] \\ \leq \text{negl}(\lambda). \end{aligned}$$

To construct  $\text{sDel}$ , we introduce a notion of *Semi-Trusted Succinct Non-Interactive Arguments*  $\text{stSNARG}$  which we formally introduce and construct in Sect. 5. After that, we prove the following lemma (cf. Lemma 11) which shows how to construct  $\text{sDel}$  using  $\text{stSNARG}$  as a building block.

**Lemma 1.** *Assuming  $T = \text{poly}(m, n)$ ,  $T, m, n \leq 2^\lambda$ , the  $\text{stSNARG}$  protocol in Fig. 2 implies the unconditional existence of a publicly verifiable non interactive succinct delegation scheme  $\text{sDel}$  as defined above.*

#### 4.1 sDel with Zero-Knowledge

A publicly verifiable non interactive succinct delegation scheme with zero knowledge  $\text{zk-sDel}$  is defined by the following efficient algorithms:

- $\text{zk-sDel.Setup}(1^\lambda)$ : A randomized setup algorithm which on input security parameter  $\lambda$  and outputs  $\text{crs}$ .
- $\text{zk-sDel.ProgAuth}(1^\lambda, \text{crs})$ : A program author which takes as input  $\lambda$ , generates a program  $P \in \{0, 1\}^m$ ,  $m \leq 2^\lambda \in \mathbb{N}$ . Additionally, it computes a digest  $H_P$  and creates a statistically binding and extractable commitment  $C_P$  of  $H_P$  under randomness  $r$ . Finally it sends a private output  $(P, \text{state})$  and public output  $C_P$ . Here  $\text{state}$  contains the randomness  $r$  and  $H_P$  encoded in it along with any other state information.
- $\text{zk-sDel.W}(\text{crs}, P, \text{state}, C_P, x)$ : A deterministic cloud worker which on input  $\text{crs}$ , program  $P$ , commitment  $C_P$ ,  $x \in \{0, 1\}^n$ ,  $n \leq 2^\lambda \in \mathbb{N}$  outputs a value  $y$  and proof  $\Pi$ .
- $\text{zk-sDel.V}(\text{crs}, x, y, C_P, \Pi)$ : A deterministic verifier which on input  $(\text{crs}, C_P, x, y, \Pi)$  either accepts or rejects.

Apart from the Completeness, Efficiency and Soundness guarantees mentioned above, a publicly verifiable succinct delegation scheme  $(\text{zk-sDel.Setup}, \text{zk-sDel.ProgAuth}, \text{zk-sDel.W}, \text{zk-sDel.V})$  satisfies the following additional property:

**Non Interactive Zero Knowledge.** For all  $\lambda, n, m \in \mathbb{N}$  such that  $n, m \leq 2^\lambda$ ,  $\forall x \in \{0, 1\}^n$  and  $y \in \{0, 1\}$ , there exists a PPT simulator  $\text{Sim} := (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$  such that the distributions of

$$(\text{crs}, x, y, C_P, \Pi) \mid (\text{crs}, \text{aux}) \leftarrow \text{Sim}_1(1^\lambda), (C_P, \text{aux}') \leftarrow \text{Sim}_2(\text{crs}, \text{aux}), \\ (y, \Pi) \leftarrow \text{Sim}_3(\text{aux}', \text{crs}, x, C_P)$$

and

$$(\text{crs}, x, y, C_P, \Pi) \mid \text{crs} \leftarrow \text{zk} - \text{sDel.Setup}(1^\lambda), ((P, \text{state}), C_P) \leftarrow \text{zk} - \text{sDel.ProgAuth}(1^\lambda, \text{crs}), \\ (y := P(x), \Pi) \leftarrow \text{zk} - \text{sDel.W}(\text{crs}, P, \text{state}, x, C_P)$$

are indistinguishable.

In Sect. 6, we present a generic construction of a semi trusted non-interactive succinct arguments with zero-knowledge (ZKstSNARG) from stSNARG. Analogous to the previous lemma, we get the following corollary (cf. Corollary 2) from Lemma 11.

**Corollary 1.** *Assuming  $T = \text{poly}(m, n)$ ,  $T, m, n \leq 2^\lambda$ , the ZKstSNARG protocol in Fig. 5 implies the unconditional existence of a publicly verifiable non interactive succinct delegation scheme with zero knowledge.*

## 5 Semi-Trusted Succinct Non-Interactive Argument (stSNARG)

We introduce a notion of “Semi-Trusted” SNARGs which is similar to the general definition of SNARGs with an addition “trusted” polynomial time algorithm that outputs a hash for the witness. Further, we provide an explicit construction of an stSNARG for all of  $NP$ . Note that any SNARG for arbitrary  $NP$  language  $\mathcal{L}$  can be reformulated as a Turing Machine which takes in as input an instance  $x$  along with witness  $w$  and accepts  $x, w$  in  $T$  steps if  $x \in \mathcal{L}$  [10]. In this work, we modify the definition of [10] by using a Universal Turing Machine  $\mathcal{TM}$  which takes as input an instance  $(x, y)$ , a witness which is a program  $P$  and accepts  $(P, x, y)$  in  $T$  steps if  $P(x) = y$ . We formalise this notion as follows:

Let  $\mathcal{TM}$  be a Universal Turing Machine which takes as input a program  $P \in \{0, 1\}^m$  for some  $m < 2^\lambda$ , and  $x \in \{0, 1\}^n$  for some  $n < 2^\lambda$  and  $y \in \{0, 1\}$  which serve as an input and output for  $P$  respectively.  $\mathcal{TM}$  accepts  $(P, x, y)$  in  $T$  steps if  $P(x) = y$ . A prover produces a proof  $\Pi$  to convince a verifier that  $\mathcal{TM}$  accepts  $P, x, y$  in  $T$ . A publicly verifiable semi-trusted SNARG (stSNARG) for  $\mathcal{TM}$  has the following polynomial time algorithms:

- $\text{stSNARG.Setup}(1^\lambda, 1^T)$ : A randomized setup algorithm which on input security parameter  $\lambda$ , and number of Turing Machine steps  $T$ , outputs  $\text{crs}$ .
- $\text{stSNARG.TrustHash}(\text{crs}, P)$ : A deterministic and honest algorithm which on input  $\text{crs}$  and a program  $P \in \{0, 1\}^m$  for some  $m < 2^\lambda$ , outputs a succinct and public digest  $H_P$  of  $P$  corresponding to  $\text{crs}$ .

- $\text{stSNARG.P}(\text{crs}, P, x, y, H_P)$ : A deterministic prover algorithm which on input the  $\text{crs}$ ,  $P \in \{0, 1\}^m$  for some  $m < 2^\lambda$ ,  $x \in \{0, 1\}^n$  for some  $n < 2^\lambda$ ,  $y \in \{0, 1\}$  and the digest  $H_P$  outputs a proof  $\Pi$ .
- $\text{stSNARG.V}(\text{crs}, x, y, H_P, \Pi)$ : A deterministic verification algorithm which on input  $\text{crs}$ ,  $x$ ,  $y$ , digest  $H_P$  and proof  $\Pi$ , either accepts(output 1) or rejects(output 0) it.

A Universal Turing Machine  $\mathcal{TM}$  on input  $(P, x, y)$  outputs 1 if it accepts  $(P, x, y)$  within  $T$  steps. We define the NP language  $\mathcal{L}_{\mathcal{TM}}$  as,

$$\mathcal{L}_{\mathcal{TM}} := \{(P, x, y, T, H_P, \text{crs}) \mid \mathcal{TM}(P, x, y) = 1 \wedge \text{stSNARG.TrustHash}(\text{crs}, P) = H_P\}.$$

Note that here  $P$  is not considered a part of the witness although it is unknown to the verifier because a typical NP statement puts a there exists constraint on the witness. In that case, the statement becomes trivial because there will always exist a program  $P$  which on input  $x$  ignores the input and outputs  $y$ . We need to ensure that  $P$  is the program output by the program author independent of  $x$ . Moreover, this is indeed a P statement for the prover.

A publicly verifiable stSNARG scheme  $\text{stSNARG} = (\text{stSNARG.Setup}, \text{stSNARG.TrustHash}, \text{stSNARG.P}, \text{stSNARG.V})$  satisfies the following properties:

- **Completeness.** For every  $\lambda, T, n, m \in \mathbb{N}$  such that  $T, n, m < 2^\lambda$ , program  $P \in \{0, 1\}^m$ , input  $x \in \{0, 1\}^n$  and output  $y \in \{0, 1\}$  such that  $(P, x, y, T, H_P, \text{crs}) \in \mathcal{L}_{\mathcal{TM}}$ , we have

$$\Pr[\text{stSNARG.V}(\text{crs}, x, y, H_P, \Pi) = 1 \mid \text{crs} \leftarrow \text{stSNARG.Setup}(1^\lambda, 1^T), \\ H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P), \Pi \leftarrow \text{stSNARG.P}(\text{crs}, P, x, y, H_P)] = 1.$$

- **Efficiency.**  $\text{stSNARG.Setup}$  runs in time  $\text{poly}(\lambda, T)$ ,  $\text{stSNARG.TrustHash}$  runs in time  $\text{poly}(\lambda, |P|, T)$ ,  $\text{stSNARG.P}$  runs in time  $\text{poly}(\lambda, |x|, |P|, T)$  and outputs a proofs of length  $\text{poly}(\lambda, \log T)$ , and  $\text{stSNARG.V}$  runs in time  $\text{poly}(\lambda, \log T)$ .
- **Soundness.** For every PPT adversary  $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$  and the tuple  $T = T(\lambda), n = n(\lambda), m = m(\lambda)$ , there exists a negligible function  $\text{negl}(\lambda)$  such that for every  $\lambda \in \mathbb{N}$ ,

$$\Pr[\text{stSNARG.V}(\text{crs}, x, y, H_P, \Pi) = 1 \wedge (P, x, y, T, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \mid \\ \text{crs} \leftarrow \text{stSNARG.Setup}(1^\lambda, 1^T), (P, \text{aux}) \leftarrow \mathcal{A}_1(1^\lambda, \text{crs}), \\ H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P), (x, y, \Pi) \leftarrow \mathcal{A}_2(\text{crs}, P, H_P, \text{aux})] \leq \text{negl}(\lambda).$$

**Protocol 1 (Semi-Trusted SNARG)**

- $\text{stSNARG.Setup}(1^\lambda, 1^T)$  :
  - $\text{SE}.K_{\text{even}} \leftarrow \text{SE.Gen}(1^\lambda, 1^{M_{\lambda,T}}, 1^{L_\lambda})^a$
  - $\text{SE}.K_{\text{odd}} \leftarrow \text{SE.Gen}(1^\lambda, 1^M, 1^L)$
  - $\text{BARG.crs} \leftarrow \text{BARG.Gen}(1^\lambda, 1^{T+1}, 1^{|C_{\text{index}}|})$
  - $\text{dk} \leftarrow \text{HT.Gen}(1^\lambda)$
  - *return*  $\text{crs} := (\text{SE}.K_{\text{even}}, \text{SE}.K_{\text{odd}}, \text{BARG.crs}, \text{dk})$ .
- $\text{stSNARG.TrustHash}(\text{crs}, P)$ 
  - $(\text{tree}_0^2, \text{rt}_0^2) \leftarrow \text{HT.Hash}(\text{dk}, P)$ ,  $H_P \leftarrow \text{rt}_0^2$
  - *return*  $H_P$ .
- $\text{stSNARG.P}(\text{crs}, P, x, y, H_P)$  :
  - $\square := \text{empty string}$
  - $(\text{tree}_0^1, \text{rt}_0^1) \leftarrow \text{HT.Hash}(\text{dk}, x)$ ,  $(\text{tree}_0^2, \text{rt}_0^2) \leftarrow \text{HT.Hash}(\text{dk}, P)$ ,  
 $(\text{tree}_0^3, \text{rt}_0^3) \leftarrow \text{HT.Hash}(\text{dk}, \square)$
  - *initialize*  $s$  with the start state of  $\mathcal{TM}$
  - $\text{st}_0 := (0, 0, 0, s)$
  - $\text{h}_0 := (\text{st}_0, \text{rt}_0^1, \text{rt}_0^2, \text{rt}_0^3)$
  - *for every*  $i = 1$  *to*  $T$ ,
    - $\text{rt}_i^1 \leftarrow \text{rt}_{i-1}^1, \text{rt}_i^2 \leftarrow \text{rt}_{i-1}^2$
    - $(l_i^1, l_i^2, l_i^3) \leftarrow \text{StepR}(\text{st}_{i-1})$
    - $\{b_i^j, \Pi_i^j\} \leftarrow \text{HT.Read}(\text{tree}_{i-1}^j, l_i^j)\}_{j \in [3]}$
    - $(b_i^j, l_i^j, \text{st}_i) \leftarrow \text{StepW}(\text{st}_{i-1}, b_i^1, b_i^2, b_i^3)$
    - $(\text{tree}_i^3, \text{rt}_i^3, \Pi_i^j) \leftarrow \text{HT.Write}(\text{tree}_{i-1}^3, l_i^3, b_i^3)$
    - $\text{h}_i \leftarrow (\text{st}_i, \text{rt}_i^1, \text{rt}_i^2, \text{rt}_i^3)$
  - $A := (\text{h}_0, (\text{h}_1, \{b_1^j, \Pi_1^j\}_{j \in [3]}, \Pi_1^j), \dots, (\text{h}_T, \{b_T^j, \Pi_T^j\}_{j \in [3]}, \Pi_T^j))$
  - $c_{\text{even}} \leftarrow \text{SE.Hash}(\text{SE}.K_{\text{even}}, A)$  and  $c_{\text{odd}} \leftarrow \text{SE.Hash}(\text{SE}.K_{\text{odd}}, A)$
  - $c := (c_{\text{even}}, c_{\text{odd}})$
  - $I_x \leftarrow \{[i_1, i_2] \mid A[i_1, i_2] = x\}$
  - $\rho_{\text{h}_0} \leftarrow \text{SE.Open}(\text{SE}.K_{\text{even}}, A, I_{\text{h}_0})^b$
  - *for every*  $i \leq \lfloor [T/2] \rfloor$ ,  
 $\text{for } B \in \{\text{h}_{2i}, \{b_{2i}^j, \Pi_{2i}^j\}_{j \in [3]}, \Pi_{2i}^j\}$ ,  $\rho_B \leftarrow \text{SE.Open}(\text{SE}.K_{\text{even}}, A, I_B)$
  - *for every*  $i \leq \lfloor [T/2] \rfloor$ ,  
 $\text{for } B \in \{\text{h}_{2i+1}, \{b_{2i+1}^j, \Pi_{2i+1}^j\}_{j \in [3]}, \Pi_{2i+1}^j\}$ ,  $\rho_B :=$   
 $\text{SE.Open}(\text{SE}.K_{\text{odd}}, A, I_B)$
  - *Let*  $C_{\text{index}}$  *be as defined in Figure 3*
  - $\Pi := \text{BARG.P}(\text{crs}, C_{\text{index}}, \text{h}_0, \{\text{h}_{i-1}, \text{h}_i, \{b_i^j, \Pi_i^j\}_{j \in [3]}, \Pi_i^j, \rho_{\text{h}_{i-1}}, \rho_{\text{h}_i},$   
 $\{\rho_{b_i^j}, \rho_{\Pi_i^j}\}_{j \in [3]}, \rho_{\Pi_i^j}\}_{i \in [T]})$
  - *return*  $(c, \Pi)$  <sup>c</sup>.
- $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi))$  :
  - *Compute*  $C_{\text{index}}$
  - *return* 1 *if and only if*  $\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1$ .

<sup>a</sup> $M_{\lambda,T} = O(T \text{ poly}(\lambda))$  and  $L_\lambda = O(\text{poly}(\lambda))$  are arbitrary and efficiently computable values which can be fixed in advance and hardcoded to the Setup algorithm during instantiation.

<sup>b</sup>Note that for simplification, we abuse notation here by specifying opening to more than a single bit.

<sup>c</sup>We often abuse notation and use  $(c, \Pi)$  to denote a proof. This can be done without loss of generalization by defining a new proof  $\Pi' = (c \parallel \Pi)$ .

**Fig. 2.** Semi-Trusted SNARG

**Circuit 1 (Circuit  $C_{\text{index}}$ )**

- **Hard-coded:**  $y, c, \text{start}, \phi, \text{SE}.K_{\text{even}}, \text{SE}.K_{\text{odd}}, T, H_P, H_x := \text{HT}.\text{Hash}(\text{dk}, x)$
- **Input:**
  - $(i, (h_i := (\text{st}_i, \text{rt}_i^1, \text{rt}_i^2, \text{rt}_i^3), \rho_{h_i}))$ , if  $i = 0$
  - $(i, (\{h_{i-1}, h_i, \{b_i^j, \Pi_i^j\}_{j \in [3]}, \Pi'_i, \rho_{h_{i-1}}, \rho_{h_i}, \{\rho_{b_i^j}, \rho_{\Pi_i^j}\}_{j \in [3]}, \rho_{\Pi'_i}\}))$ ,  $\forall i \in [T]$
- **Output:** return 1 if and only if
  - if  $i = 0$ 
    - a.  $\text{st}_0 = \text{start}$
    - b.  $H_x = \text{rt}_0^1$
    - c.  $H_P = \text{rt}_0^2$
    - d.  $\text{HT}.\text{Hash}(\text{dk}, \square)$  has  $\text{rt}_0^3$  as root
  - else
    - \* if  $i$  is even:
      - a.  $\text{SE}.\text{Verify}(\text{SE}.K_{\text{odd}}, C_{\text{odd}}, h_{i-1}, \rho_{h_{i-1}}) = 1$
      - b.  $\text{SE}.\text{Verify}(\text{SE}.K_{\text{even}}, C_{\text{even}}, h_i, \rho_{h_i}) = 1$
      - c.  $\left\{ \text{SE}.\text{Verify}(\text{SE}.K_{\text{even}}, C_{\text{even}}, b_i^j, \rho_{b_i^j}) = 1 \right\}_{j \in [3]}$
      - d.  $\left\{ \text{SE}.\text{Verify}(\text{SE}.K_{\text{even}}, C_{\text{even}}, \Pi_i^j, \rho_{\Pi_i^j}) = 1 \right\}_{j \in [3]}$
      - e.  $\text{SE}.\text{Verify}(\text{SE}.K_{\text{even}}, C_{\text{even}}, \Pi'_i, \rho_{\Pi'_i}) = 1$
    - \* if  $i$  is odd:
      - a.  $\text{SE}.\text{Verify}(\text{SE}.K_{\text{even}}, C_{\text{even}}, h_{i-1}, \rho_{h_{i-1}}) = 1$
      - b.  $\text{SE}.\text{Verify}(\text{SE}.K_{\text{odd}}, C_{\text{odd}}, h_i, \rho_{h_i}) = 1$
      - c.  $\left\{ \text{SE}.\text{Verify}(\text{SE}.K_{\text{odd}}, C_{\text{odd}}, b_i^j, \rho_{b_i^j}) = 1 \right\}_{j \in [3]}$
      - d.  $\left\{ \text{SE}.\text{Verify}(\text{SE}.K_{\text{odd}}, C_{\text{odd}}, \Pi_i^j, \rho_{\Pi_i^j}) = 1 \right\}_{j \in [3]}$
      - e.  $\text{SE}.\text{Verify}(\text{SE}.K_{\text{odd}}, C_{\text{odd}}, \Pi'_i, \rho_{\Pi'_i}) = 1$
    - \*  $\phi(h_{i-1}, h_i, \{b_i^j, \Pi_i^j\}_{j \in [3]}, \Pi'_i) = 1$
    - \* if  $i = T$ 
      - a.  $\text{HT}.\text{Hash}(\text{dk}, y)$  has  $\text{rt}_T^3$  as root.
      - b.  $\text{st}_T$  indeed encodes the accept state.

**Fig. 3.** Circuit  $C_{\text{index}}$ **5.1 Our Construction**

Our construction is formulated similar to that of [10]. Specifically, we use the notion of non-interactive BARG for index language and SE Hash functions in our scheme.

*Setup for Universal Turing Machine.* For a cleaner analysis, we assume without loss of generality that  $\mathcal{TM}$  consists of three tapes, namely,  $\text{Tp}_1, \text{Tp}_2, \text{Tp}_3$ .  $\text{Tp}_1$  and  $\text{Tp}_2$  are read only tapes that store  $x$  and  $P$  respectively.  $\text{Tp}_3$  is the work tape which is initialized with  $\square$  to denote an empty string.

*Transition steps for  $\mathcal{TM}$ .*  $\mathcal{TM}$ 's state information along with the head locations of the three tapes are encoded as  $\text{st}$ . To handle Turing Machines with arbitrarily long tapes, we encode  $\{\text{Tp}_i\}_{i \in [3]}$  using three Hash Trees as defined in previous sections and produce tree roots  $\text{rt}^1, \text{rt}^2, \text{rt}^3$  respectively.

Let the each intermediate transition state of  $\mathcal{TM}$  be encoded as  $h_i := (\text{st}_i, \text{rt}_i^1, \text{rt}_i^2, \text{rt}_i^3)$  for  $i \in [T]$ . A single step of  $\mathcal{TM}$  can be interpreted in the manner described below which is similar to one described for a RAM in [23]. We break down the step function at the  $i^{\text{th}}$  stage into two deterministic polynomial time algorithms:

- **StepR:** On input  $\text{st}_{i-1}$  of  $\mathcal{TM}$ , outputs head positions  $l_{i-1}^1, l_{i-1}^2, l_{i-1}^3$  which denote the memory locations of  $\text{Tp}_1, \text{Tp}_2, \text{Tp}_3$  which  $\mathcal{TM}$  in the current state  $\text{st}_{i-1}$  would read from.
- **StepW:** On input  $\text{st}_{i-1}$ , and bits  $b_{i-1}^1, b_{i-1}^2, b_{i-1}^3$  outputs bit  $b'$ , location  $l'$  and  $\text{st}_i$  such that  $\mathcal{TM}$  upon reading  $b_{i-1}^1, b_{i-1}^2, b_{i-1}^3$  at locations  $l_{i-1}^1, l_{i-1}^2, l_{i-1}^3$  using  $\text{HT.Read}$ , would write  $b'$  at location  $l'$  of  $\text{Tp}_3$ , thereby transition to new state  $\text{st}_i$ .

Now, we translate the  $i^{\text{th}}$  single step of  $\mathcal{TM}$  to the circuit  $\phi$  which is defined such that on input digests  $h_{i-1} := (\text{st}_{i-1}, \text{rt}_{i-1}^1, \text{rt}_{i-1}^2, \text{rt}_{i-1}^3)$  and  $h_i := (\text{st}_i, \text{rt}_i^1, \text{rt}_i^2, \text{rt}_i^3)$ , bits  $b_i^1, b_i^2, b_i^3$ , and proofs  $\Pi_i^1, \Pi_i^2, \Pi_i^3, \Pi'_i$ ,  $\phi(h_{i-1}, h_i, b_i^1, b_i^2, b_i^3, \Pi_i^1, \Pi_i^2, \Pi_i^3, \Pi'_i) = 1$  if and only if the following hold:

- |  |   |
|--|---|
| <ol style="list-style-type: none"> <li>1. <math>(l_i^1, l_i^2, l_i^3) \leftarrow \text{StepR}(\text{st}_{i-1})</math></li> <li>2. <math>(b', l', \text{st}') \leftarrow \text{StepW}(\text{st}_{i-1}, b_i^1, b_i^2, b_i^3)</math></li> <li>3. <math>\text{st}' = \text{st}_i</math></li> <li>4. <math>\text{HT.VerRead}(\text{dk}, \text{rt}_{i-1}^1, l_i^1, b_i^1, \Pi_i^1) = 1</math></li> <li>5. <math>\text{HT.VerRead}(\text{dk}, \text{rt}_{i-1}^2, l_i^2, b_i^2, \Pi_i^2) = 1</math></li> </ol> | <ol style="list-style-type: none"> <li>6. <math>\text{HT.VerRead}(\text{dk}, \text{rt}_{i-1}^3, l_i^3, b_i^3, \Pi_i^3) = 1</math></li> <li>7. <math>\text{rt}_i^1 = \text{rt}_{i-1}^1</math></li> <li>8. <math>\text{rt}_i^2 = \text{rt}_{i-1}^2</math></li> <li>9. <math>\text{HT.VerWrite}(\text{dk}, \text{rt}_{i-1}^3, l', b', \text{rt}_i^3, \Pi'_i) = 1</math></li> </ol> |
|--|---|

Here,  $\text{dk}$  denote the hash keys used to build the three hash trees. Note that the efficiency of hash tree implies that  $\phi$  can be constructed such that it can be represented as a formula in  $L = \text{poly}(\lambda)$  variables. For the  $T$  steps of  $\mathcal{TM}$ , we have the following formula over  $M = O(L \cdot T)$  variables:

$$\Phi(\mathbf{h}_0, \{h_i, b_i^1, b_i^2, b_i^3, \Pi_i^1, \Pi_i^2, \Pi_i^3, \Pi'_i\}_{i \in [T]}) = \bigwedge_{i \in [T]} \phi(h_{i-1}, h_i, b_i^1, b_i^2, b_i^3, \Pi_i^1, \Pi_i^2, \Pi_i^3, \Pi'_i)$$

Following the techniques in [10], we use a combination of SE Hash along with  $\phi$  to produce the circuit for index languages.

Our semi-trusted SNARG scheme is given in Fig. 2 and the corresponding index language circuit is shown as Fig. 3.

**Theorem 3.** *Assuming the existence of Somewhere Extractable Hash functions, non-interactive Batch Arguments for Index Languages, and Collision Resistant Hash Trees as described in Sect. 3, Fig. 2 is a publicly verifiable non-interactive semi-trusted SNARG with CRS size, proof size and verifier time  $\text{poly}(\lambda, \log T)$  and prover run time being  $\text{poly}(\lambda, T)$ .*

*Completeness.* Here we give a sketch arguing completeness of our scheme. Our construction in Fig. 2 tells that

$$\begin{aligned} & \Pr[\text{stSNARG.V}(\text{crs}, x, y, H_P, \Pi) = 1 \mid \text{crs} \leftarrow \text{stSNARG.Setup}(1^\lambda, 1^T), \\ & H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P), \Pi \leftarrow \text{stSNARG.P}(\text{crs}, P, x, y, H_P)] = \\ & \Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \mid \text{crs} \leftarrow \text{stSNARG.Setup}(1^\lambda, 1^T), \\ & H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P), \Pi \leftarrow \text{stSNARG.P}(\text{crs}, P, x, y, H_P)] \end{aligned}$$

where  $C_{\text{index}}$  is the index circuit as shown in Fig. 3. Observing  $\text{stSNARG.P}$  algorithm in our scheme tells it is sufficient to show that if the prover is honest and uses a valid witness, then  $(C_{\text{index}}, i) \in \mathcal{L}_{\text{index}}, \forall i \in \{0\} \cup [T]$ . If we can argue that this is indeed the case, then the completeness of BARG gives the desired result.

If  $(P, x, y, T, H_P, \text{crs}) \in \mathcal{L}_{\mathcal{TM}}$ , then  $(C_{\text{index}}, 0) \in \mathcal{L}_{\text{index}}$  is trivially true by observation. Now, let us look at  $(C_{\text{index}}, 1)$ . We start by analysing that  $\phi(\text{h}_0, \text{h}_1, \{b_1^j, \Pi_1^j\}_{j \in [3]}, \Pi_1^1) = 1$  is true.  $\{\text{rt}_1^i = \text{rt}_0^i\}_{i \in [2]}$  follow from the read-only nature of tapes  $\text{Tp}_1, \text{Tp}_2$ . Since,  $\left\{ (b_1^j, \Pi_1^j) \leftarrow \text{HT.Read}(\text{tree}_0^j, l_1^j) \right\}_{j \in [3]}$ , the hash tree completeness of read ensures that  $\{\text{HT.VerRead}(\text{dk}, \text{rt}_0^i, l_1^i, b_1^i, \Pi_1^i) = 1\}_{i \in [3]} = 1$  and  $\{\text{Tp}_i[l_1^i] = b_1^i\}_{i \in [3]}$ . This along with the correctness of Turing Machine StepR function implies that  $b_1^1, b_2^1, b_3^1$  are indeed the correct input for the StepW function of  $\mathcal{TM}$ . Finally,  $(\text{tree}_1^3, \text{rt}_1^3, \Pi_1^1) \leftarrow \text{HT.Write}(\text{tree}_0^3, l_1^1, b_1^1)$  implies  $\text{HT.VerWrite}(\text{dk}, \text{rt}_0^3, l_1^1, b_1^1, \text{rt}_1^3, \Pi_1^1) = 1$  from the hash tree completeness of write property. The same property also ensures that  $\text{Tp}_3$  changes only at the  $l^{th}$  memory location. When paired with the correctness of StepW, we get that  $\text{st}_1 = \text{st}'$

The completeness of the SE hash implies that the verification algorithm certainly accepts all the local openings. Thus,  $(C_{\text{index}}, 1) \in \mathcal{L}^{\text{index}}$ . Now,  $(C_{\text{index}}, T) \in \mathcal{L}^{\text{index}}$  because  $\mathcal{TM}$  accept  $(P, x, y)$  in  $T$  steps. We can show in a similar manner that for all other  $i$ ,  $(C_{\text{index}}, i) \in \mathcal{L}^{\text{index}}$ . This proves the completeness of the scheme in Fig. 2.

*Efficiency.*

- Runtime of  $\text{stSNARG.Setup}$  is  $\text{poly}(\lambda, T)$ . This follows from the efficiency of underlying primitives.
- $\text{stSNARG.TrustHash}$  computes  $H_P$  in time  $|P| \cdot \text{poly}(\lambda)$  which is  $\text{poly}(|P|, \lambda)$ .
- $|C_{\text{index}}| = \text{poly}(\lambda, \log T)$ . This follows from the efficiency of the SE hash and the efficiency of hash tree construction.
- CRS Size: By the corresponding properties of the underlying primitives,  $|\text{crs}| = \text{poly}(\lambda, \log T)$ .
- The prover’s computation time is dominated by the hashes corresponding to  $x, P$  and the Turing Machine step functions that is run  $T$  times. This requires a total time of  $\text{poly}(\lambda, |x|) + \text{poly}(\lambda, |P|) + \text{poly}(\lambda, T) = \text{poly}(\lambda, |x|, |P|, T)$ .
- Proof Length:  $|c| + |\Pi| = \text{poly}(\lambda, \log T) + \text{poly}(\lambda, \log T, |C_{\text{index}}|) = \text{poly}(\lambda, \log T)$ .
- Verifier Time: Time taken to compute  $C_{\text{index}}$  and verify the BARG. This is  $\text{poly}(\lambda, \log T, |C_{\text{index}}|) = \text{poly}(\lambda, \log T)$ .



*Soundness.* Let us assume for the sake of contradiction that our scheme in Fig. 2 is not sound, i.e., there exists a PPT adversary  $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ , a value  $T$  and a polynomial function  $\text{poly}(\lambda)$  such that for infinitely many values of  $\lambda \in \mathbb{N}$ ,

$$\Pr[\mathbf{G}^{\mathcal{A}} = 1] \geq \frac{1}{\text{poly}(\lambda)},$$

where  $\mathcal{A}$  plays Game G described below

Real Game G

1.  $\text{crs} \leftarrow \text{stSNARG.Setup}(1^\lambda, 1^T)$
2.  $(P, \text{aux}) \leftarrow \mathcal{A}_1(1^\lambda, \text{crs})$
3.  $H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P)$
4.  $((x, y)(c, \Pi)) \leftarrow \mathcal{A}_2(\text{crs}, P, H_P, \text{aux})$
5. if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}}$ ,  
return 1
6. else return 0

Let  $S_i$  denote the following set:

$$S_i = \begin{cases} h_0 & \text{if } i = 0 \\ \{h_i, \{b_i\}_{j \in [3]}, \{\Pi_i\}_{j \in [3]}, \Pi'_i\} & \text{if } i \in [T] \end{cases}$$

Let  $D$  denote the string  $(h_0, \{h_i, \{b_i\}_{j \in [3]}, \{\Pi_i\}_{j \in [3]}, \Pi'_i\}_{i \in [T]})$ .  $I_{S_i} \subset |D|$  denotes the following:

$$I_{S_i} = \{[a, b] \mid a, b \in |D|, D[a, b] = S_i\}.$$

In game G, say we have  $(\text{tree}_0^1, \text{rt}_0^1) \leftarrow \text{HT.Hash}(\text{dk}, x)$ ,  $(\text{tree}_0^2, \text{rt}_0^2) \leftarrow \text{HT.Hash}(\text{dk}, P)$ ,  $(\text{tree}_0^3, \text{rt}_0^3) \leftarrow \text{HT.Hash}(\text{dk}, \square)$ . Also, let  $\text{st}_0 := (0, 0, 0, s)$ , where  $s$  is the start state of  $\mathcal{TM}$ . We say that  $\bar{h}_0 := (\text{st}_0, \text{rt}_0^1, \text{rt}_0^2, \text{rt}_0^3)$  defines a unique “true” digest for the starting step of  $\mathcal{TM}$ .

If  $\text{stSNARG.V}(\text{crs}, x, H_P, c, \Pi) = 1$ , then Algorithm  $\text{Step}(x, P, \text{crs}, i)$  in Fig. 4 computes the unique true digest  $\bar{h}_i$  after the  $i^{\text{th}}$  Turing Machine Step along with the other uniquely correct values of the set  $\bar{S}_i := \{\bar{h}_i, \{\bar{b}_i\}_{j \in [3]}, \{\bar{\Pi}_i\}_{j \in [3]}, \bar{\Pi}'_i\}$ . We use the notation  $\text{Step}(x, P, \text{crs}, i).x$  to denote  $x \in \bar{S}_i$ . We proceed by performing an induction on the following sequence outer hybrid games  $G_i, i$  from 1 to  $T$ . We use a sequence of inner hybrid games to transition between subsequent outer hybrids. Our induction hypothesis is that, under suitable assumptions, for all  $i \in 1$  to  $T$ , there exists a negligible function  $\lambda$  such that,

$$\Pr[\mathbf{G}^{\mathcal{A}} = 1] \leq \Pr[G_i^{\mathcal{A}} = 1] + \text{negl}(\lambda).$$

Intuitively, the  $i^{\text{th}}$  game  $G_i$  is similar to the real life soundness game with the following two changes: (1) The key generation for the SE hash and BARG is done in the trapdoor mode at the  $i^{\text{th}}$  game. This allows for extractability of the  $i^{\text{th}}$  block of the string  $D$  from the commitment  $c$ . (2) The adversary wins the game if they break the soundness assumption as the real life game G and the extracted block is indeed the correct one.

Algorithm  $Step(x, y, P, crs, i)$

- $\square :=$  empty string
- $(tree_0^1, rt_0^1) := HT.Hash(dk, x)$ ,  $(tree_0^2, rt_0^2) := HT.Hash(dk, P)$ ,  $(tree_0^3, rt_0^3) := HT.Hash(dk, \square)$
- initialize  $s$  with the start state of  $\mathcal{TM}$
- $st_0 := (0, 0, 0, s)$
- $\bar{h}_0 := (st_0, rt_0^1, rt_0^2, rt_0^3)$
- if  $i = 0$ , return  $\bar{S}_0 := (st_0, rt_0^1, rt_0^2, rt_0^3)$
- else
  - for  $count = 1$  to  $i$ ,
    - $(l_{count}^1, l_{count}^2, l_{count}^3) \leftarrow StepR(st_{count-1})$
    - $\{(b_{count}^k, \Pi_{count}^k) := HT.Read(tree_{count-1}^k, l_{count}^k)\}_{k \in [3]}$
    - $(b_{count}^3, l_{count}^3, st_{count}) := StepW(st_{count-1}, b_{count}^1, b_{count}^2, b_{count}^3)$
    - $(tree_{count}^3, rt_{count}^3, \Pi'_{count}) := HT.Write(tree_{count-1}^3, l_{count}^3, b_{count}^3)$
  - $\bar{h}_i := (st_i, rt_i^1, rt_i^2, rt_i^3)$
  - $\bar{b}_i := (b_i^1, b_i^2, b_i^3)$
  - $\bar{l}_i := (l_i^1, l_i^2, l_i^3)$
  - $\bar{r}_i := (rt_{i-1}^1, rt_{i-1}^2, rt_i^3)$
  - $\bar{\Pi}_i := (\Pi_i^1, \Pi_i^2, \Pi_i^3, \Pi'_i)$
  - return  $\bar{S}_i := (\bar{h}_i, \bar{b}_i, \bar{r}_i, \bar{\Pi}_i)$

Fig. 4. Turing Machine  $i^{th}$  step.

Outer Hybrid Game  $G_i$

1. if  $i$  is even
  - $SE.K_{even} \leftarrow SE.TGen(1^\lambda, 1^M, I_{S_i})$
  - $SE.K_{odd} \leftarrow SE.TGen(1^\lambda, 1^M, I_{S_{i-1}})$
2. if  $i$  is odd
  - $SE.K_{even} \leftarrow SE.TGen(1^\lambda, 1^M, I_{S_{i-1}})$
  - $SE.K_{odd} \leftarrow SE.TGen(1^\lambda, 1^M, I_{S_i})$
3.  $BARG.crs \leftarrow BARG.TGen(1^\lambda, 1^{T+1}, 1^{C_{index}}, i)$
4.  $dk \leftarrow HT.Gen(1^\lambda)$
5.  $crs := (SE.K_{even}, SE.K_{odd}, BARG.crs, dk)$ .
6.  $(P, aux) \leftarrow \mathcal{A}_1(1^\lambda, crs)$
7.  $H_P \leftarrow stSNARG.TrustHash(crs, P)$
8.  $((x, y), (c, \Pi)) \leftarrow \mathcal{A}_2(crs, P, aux)$
9. Parse  $c$  as  $(c_{odd}, c_{even})$
10. if  $i$  is even and  $i \neq 0$ 
  - $(h_i, \{b_i^k\}_{k \in [3]}, \{\Pi_i^k\}_{k \in [3]}, \Pi'_i) \leftarrow SE.Ext_{even}(c_{even}, SE.K_{even})$
  - $(h_{i-1}, \{b_{i-1}^k\}_{k \in [3]}, \{\Pi_{i-1}^k\}_{k \in [3]}, \Pi'_{i-1}) \leftarrow SE.Ext_{odd}(c_{odd}, SE.K_{odd})$
11. if  $i$  is odd
  - $(h_i, \{b_i^k\}_{k \in [3]}, \{\Pi_i^k\}_{k \in [3]}, \Pi'_i) \leftarrow SE.Ext_{odd}(c_{odd}, SE.K_{odd})$
  - if  $i - 1 > 0$  then  $(h_{i-1}, \{b_{i-1}^k\}_{k \in [3]}, \{\Pi_{i-1}^k\}_{k \in [3]}, \Pi'_{i-1}) \leftarrow SE.Ext_{even}(c_{even}, SE.K_{even})$
  - if  $i - 1 = 0$  then  $h_0 \leftarrow SE.Ext_{even}(c_{even}, SE.K_{even})$

12. if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{T}, \mathcal{M}} \wedge h_i = \text{Step}(x, y, P, \text{crs}, i) \cdot \bar{h}_i$ , return 1
13. else return 0

*Base Case:* Assuming key indistinguishability and soundness of SE hash and BARG, we need to show that  $\Pr[G^A = 1] \leq \Pr[G_1^A = 1] + \text{negl}(\lambda)$ .

We begin by using a sequence of hybrids to transition from  $G$  to an intermediate game  $G_0$ . The colored texts in the hybrids below indicate the steps in the hybrids exclusively appear in a particular game. We only present proof sketches for the intermediate lemmas in this section due to lack of space. Concrete proofs have been shifted to the Supplementary Material.

Hybrid Games  $G_a, G_b, G_{ab}, G_0$

1.  $\text{SE.K}_{\text{even}} \leftarrow \text{SE.TGen}(1^\lambda, 1^M, I_{S_0}) \dots (G_a, G_b, G_{ab}, G_0)$
2.  $\text{SE.K}_{\text{odd}} \leftarrow \text{SE.Gen}(1^\lambda, 1^M) \dots (G_a)$
3.  $\text{SE.K}_{\text{odd}} \leftarrow \text{SE.TGen}(1^\lambda, 1^M, I_{S_1}) \dots (G_b, G_{ab}, G_0)$
4.  $\text{BARG.crs} \leftarrow \text{BARG.Gen}(1^\lambda, 1^{T+1}, 1^{|C_{\text{index}}|}) \dots (G_a, G_b)$
5.  $\text{BARG.crs} \leftarrow \text{BARG.TGen}(1^\lambda, 1^{T+1}, 1^{|C_{\text{index}}|}, 0) \dots (G_{ab}, G_0)$
6.  $\text{dk} \leftarrow \text{HT.Gen}(1^\lambda)$
7.  $\text{crs} := (\text{SE.K}_{\text{even}}, \text{SE.K}_{\text{odd}}, \text{BARG.crs}, \text{dk})$ .
8.  $(P, \text{aux}) \leftarrow \mathcal{A}_1(1^\lambda, \text{crs})$
9.  $H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P)$
10. if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{T}, \mathcal{M}}$ , return 1
11.  $h_0 \leftarrow \text{SE.Ext}_{\text{even}}(c_{\text{even}}, \text{SE.K}_{\text{even}}) \dots (G_0) \dots (G_a, G_b, G_{ab})$
12. if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{T}, \mathcal{M}} \wedge h_0 = \text{Step}(x, y, P, \text{crs}, 0) \cdot \bar{h}$ , return 1  $\dots (G_0)$
13. else return 0

**Lemma 2.** *Assuming key indistinguishability of SE,  $|\Pr[G^A = 1] - \Pr[G_a^A = 1]| \leq \text{negl}(\lambda)$ .*

*Proof.* The only difference in Game  $G$  and  $G_a$  is that the key generation algorithm of the SE hash ( $\text{SE.Gen}$ ) is replaced by the trapdoor key generation ( $\text{SE.TGen}$ ).

If  $|\Pr[G^A = 1] - \Pr[G_a^A = 1]| > \text{negl}(\lambda)$ , then one can construct a PPT adversary  $\mathcal{B}$  that breaks the key indistinguishability of SE using  $I_{S_0}$  with  $\text{Key}$  as input from the key generation algorithm of the SE hash and runs  $\mathcal{A}$  on  $\text{Key}$ . Here,  $\text{Key}$  is either  $\text{SE.Gen}(1^\lambda, 1^M)$  or  $\text{SE.TGen}(1^\lambda, 1^M, I_{S_0})$  based on whether  $\mathcal{A}$  is interacting with game  $G$  or  $G_a$  respectively. Note that the reduction can simulate the other steps of games  $G$  or  $G_a$ . Now, the probability that  $\mathcal{B}$  returns 1 in either case is exactly equal to the probability that  $\mathcal{A}$  wins the corresponding games, hence,  $\mathcal{B}$  breaks if  $|\Pr[G^A = 1] - \Pr[G_a^A = 1]| \geq \text{negl}(\lambda)$ . This leads to a contradiction of our assumption.

**Lemma 3.** *Assuming key indistinguishability of SE,  $|\Pr[G_a^A = 1] - \Pr[G_b^A = 1]| \leq \text{negl}(\lambda)$ .*

This again follows from the key-indistinguishability of SE as shown in the previous lemma as the only difference in these games is that the key generation algorithm for the SE hash has been changed to TGen, hence we skip the proof.

**Lemma 4.** *Assuming key indistinguishability of BARG,  $|\Pr[G_b^A = 1] - \Pr[G_{ab}^A = 1]| \leq \text{negl}(\lambda)$ .*

*Proof.* The only difference in Game  $G_b$  and  $G_{ab}$  is that the key generation algorithm of the BARG (BARG.Gen) is replaced by the trapdoor key generation (BARG.TGen) at index 0.

If  $|\Pr[G_b^A = 1] - \Pr[G_{ab}^A = 1]| > \text{negl}(\lambda)$ , then one can construct a PPT adversary  $\mathcal{B}$  getting Key as input that breaks the key indistinguishability of BARG, where Key is either BARG.Gen( $1^\lambda, 1^{T+1}, 1^{|C_{\text{index}}|}$ ) or BARG.TGen( $1^\lambda, 1^{T+1}, 1^{|C_{\text{index}}|}, 0$ ) based on whether  $\mathcal{A}$  is interacting with game  $G_b$  or  $G_{ab}$  respectively. The reduction then following in a similar manner as the SE key indistinguishability adversary described above.

**Lemma 5.** *Assuming somewhere soundness of BARG,*

$$|\Pr[G_{ab}^A = 1] - \Pr[G_0^A = 1]| \leq \text{negl}(\lambda).$$

*Proof.* The only difference in Games  $G_{ab}$  and  $G_0$  is that there is an additional step which computes the true digest at index 0 and extracts at the  $0^{\text{th}}$  index from  $c_{\text{even}}$  using the extraction function of SE. Finally, the adversary wins if and only if the extracted value matches the true digest along with the usual win conditions in the previous game.

Note that,

$$\begin{aligned} |\Pr[G_{ab}^A = 1] - \Pr[G_0^A = 1]| &\leq \Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \wedge \\ &\quad ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{T}, \mathcal{M}} \wedge h_0 \neq \text{Step}(x, P, \text{crs}, 0).\bar{h}] \leq \\ &\Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \wedge h_0 \neq \text{Step}(x, P, \text{crs}, 0).\bar{h}]. \end{aligned}$$

Let us assume that there exists a PPT adversary  $\mathcal{A}$  such that for infinitely many values of  $\lambda \in \mathbb{N}$ ,

$$\Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \wedge h_0 \neq \text{Step}(x, y, P, \text{crs}, 0).\bar{h}] \geq \frac{1}{\text{poly}(\lambda)}.$$

Notice that  $h_0 \neq \text{Step}(x, P, \text{crs}, 0).\bar{h}$  implies that at least one of the conditions  $\text{st}_0 = \text{start}$ ,  $H_x = \text{rt}_0^1$ ,  $H_P = \text{rt}_0^2$  and  $\text{HT.Hash}(\text{dk}, \square)$  having  $\text{rt}_0^3$  as root must not be true. If this is indeed true then our construction of  $C_{\text{index}}$  in Fig. 3 implies that  $(C_{\text{index}}, 0) \notin \mathcal{L}^{\text{index}}$ .

We now construct the following PPT adversary  $\mathcal{B}$  playing the semi-adaptive somewhere soundness game of the BARG as follows

Adversary  $\mathcal{B}$  playing semi adaptive somewhere soundness game of BARG.

- $\text{SE}.K_{\text{even}} \leftarrow \text{SE.TGen}(1^\lambda, 1^M, I_{S_0})$
- $\text{SE}.K_{\text{odd}} \leftarrow \text{SE.TGen}(1^\lambda, 1^M, I_{S_1})$

- $\text{BARG.crs} \leftarrow \text{BARG.TGen}(1^\lambda, 1^{T+1}, 1^{|C_{\text{index}}|}, 0)$
- $\text{dk} \leftarrow \text{HT.Gen}(1^\lambda)$
- $\text{crs} := (\text{SE}.K_{\text{even}}, \text{SE}.K_{\text{odd}}, \text{BARG.crs}, \text{dk})$ .
- $(P, \text{aux}) \leftarrow \mathcal{A}_1(1^\lambda, \text{crs})$
- $H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P)$
- $((x, y), (c, \Pi)) \leftarrow \mathcal{A}_2(\text{crs}, P, \text{aux})$
- return  $(C_{\text{index}}, \Pi)$

By our assumption, it is clear that  $\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1$  with non negligible probability but  $(C_{\text{index}}, 0) \notin \mathcal{L}^{\text{index}}$ . Thus,  $\mathcal{B}$  will break the semi-adaptive somewhere soundness of BARG at index 0. Therefore, it must be the case that for every PPT adversary  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\lambda)$  such that for all  $\lambda \in \mathbb{N}$ ,

$$\Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \wedge h_0 \neq \text{Step}(x, y, P, \text{crs}, 0).\bar{h}] \leq \text{negl}(\lambda)$$

$$\implies |\Pr[G_{ab}^{\mathcal{A}} = 1] - \Pr[G_0^{\mathcal{A}} = 1]| \leq \text{negl}(\lambda)$$

Now, we transition from  $G_0$  to the base case for our induction,  $G_1$  using the following sequence of indistinguishable hybrids:

$G_{0,a}$  Identical to  $G_0$  except we add an extraction:  $(h_1, \{b_1^k\}_{k \in [3]}, \{\Pi_1^k\}_{k \in [3]}, \Pi'_i) \leftarrow \text{SE.Ext}_{\text{odd}}(c_{\text{odd}}, \text{SE}.K_{\text{odd}})$  which is not used in the hybrid, hence indistinguishability follows.

$G_{0,b}$  The BARG key generation's trapdoor is changed from 0 to 1. This can be done due to key indistinguishability of BARG.

$G_{0,c}$  The winning condition is changed to: if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \wedge h_0 = \text{Step}(x, y, P, \text{crs}, 0).\bar{h} \wedge \{b_1^k\}_{k \in [3]} = \text{Step}(x, y, P, \text{crs}, 1).\bar{b} \wedge \{rt_1^k\}_{k \in [3]} = \text{Step}(x, y, P, \text{crs}, 1).\bar{rt} \wedge \text{st}_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{st}}$ , return 1

$G_{0,d}$  The winning condition is changed to: if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \wedge h_0 = \text{Step}(x, y, P, \text{crs}, 0).\bar{h} \wedge \{b_1^k\}_{k \in [3]} = \text{Step}(x, y, P, \text{crs}, 1).\bar{b} \wedge \text{rt}_1^3 = \text{Step}(x, y, P, \text{crs}, 1).\bar{rt} \wedge \text{st}_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{st}} \wedge h_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{h}$ , return 1

$G_{0,e}$  The winning condition is changed to: if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \wedge h_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{h}$ , return 1

The indistinguishability of the last three hybrids follow from the following lemmas.

**Lemma 6.** *Assuming semi-adaptive somewhere soundness of BARG, extraction correctness of SE, read and write soundness of HT,*

$$|\Pr[G_{0,b}^{\mathcal{A}} = 1] - \Pr[G_{0,c}^{\mathcal{A}} = 1]| \leq \text{negl}(\lambda).$$

*Proof.* The only difference in Games  $G_{0,b}$  and  $G_{0,c}$  is that we have added some additional conditions for the adversary to win along with the ones in the previous game.

Note that,

$$\begin{aligned} |\Pr[G_{0,b}^A = 1] - \Pr[G_{0,c}^A = 1]| &\leq \Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \wedge \\ &\quad \mathbf{h}_0 = \text{Step}(x, P, \text{crs}, 0) \cdot \bar{\mathbf{h}} \wedge \left( \{b_1^k\}_{k \in [3]} \neq \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\mathbf{b}} \right. \\ &\quad \left. \vee \{\mathbf{rt}_1^k\}_{k \in [3]} \neq \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\mathbf{rt}} \vee \mathbf{st}_1 = \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\mathbf{st}} \right)]. \end{aligned}$$

Let us assume that there exists a PPT adversary  $\mathcal{A}$  such that for infinitely many values of  $\lambda \in \mathbb{N}$ ,

$$\begin{aligned} \Pr[\text{BARG.V}(\text{BARG.crs}, C_{\text{index}}, \Pi) = 1 \wedge \mathbf{h}_0 = \text{Step}(x, y, P, \text{crs}, 0) \cdot \bar{\mathbf{h}} \\ \wedge \left( \{b_1^k\}_{k \in [3]} \neq \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\mathbf{b}} \vee \{\mathbf{rt}_1^k\}_{k \in [3]} \neq \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\mathbf{rt}} \right. \\ \left. \vee \mathbf{st}_1 = \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\mathbf{st}} \right)] \geq \frac{1}{\text{poly}(\lambda)}. \end{aligned}$$

Notice that  $\mathbf{h}_0 = \text{Step}(x, P, \text{crs}, 0) \cdot \bar{\mathbf{h}}$  implies that the conditions  $\mathbf{st}_0 = \text{start}$ ,  $H_x = \mathbf{rt}_0^1$ ,  $H_P = \mathbf{rt}_0^2$  and  $\text{HT.Hash}(\text{dk}, \square)$  having  $\mathbf{rt}_0^3$  as root are true. In other words,  $\mathbf{h}_0$  is indeed the true digest at step 0.

Assuming extraction correctness of SE, read and write soundness of HT, we construct the following PPT adversary  $\mathcal{B}$  playing the semi-adaptive somewhere soundness game of the BARG similar to the one in the proof of Lemma 5.

Thus,  $\mathcal{B}$  will break the semi-adaptive somewhere soundness of BARG at index 1 if  $(C_{\text{index}}, 1) \notin \mathcal{L}^{\text{index}}$ .

It is now left to show that  $(C_{\text{index}}, 1) \notin \mathcal{L}^{\text{index}}$ .

Case 1 If the SE verifications in  $C_{\text{index}}$  do not all return 1, then by construction of  $C_{\text{index}}$ , we have that  $(C_{\text{index}}, 1) \notin \mathcal{L}^{\text{index}}$ .

Case 2 All SE verifications return 1. Extraction Correctness/ Somewhere binding property of SE hash implies that  $\mathbf{h}_0 = (\mathbf{st}_0, \mathbf{rt}_0^1, \mathbf{rt}_0^2, \mathbf{rt}_0^3)$ ,  $\mathbf{h}_1$ ,  $\{b_1^k, \Pi_1^k\}_{k \in [3]}$ ,  $\Pi_1'$  were indeed committed by the prover as the Turing machine output at step 0 and step 1. Now, let us analyze  $\phi(\mathbf{h}_0, \mathbf{h}_1, \{b_1^k, \Pi_1^k\}_{k \in [3]}, \Pi_1')$ . By assumption, we know that  $\mathbf{h}_0 = \bar{\mathbf{h}}_0$ , i.e.,  $\bar{\mathbf{st}}_0, \bar{\mathbf{rt}}_0^1, \bar{\mathbf{rt}}_0^2, \bar{\mathbf{rt}}_0^3 = \mathbf{st}_0, \mathbf{rt}_0^1, \mathbf{rt}_0^2, \mathbf{rt}_0^3$ . StepR being a deterministic function ensures that  $(l_1^1, l_1^2, l_1^3)$  are indeed the correct Turing machine memory locations to be read at step 1. Thus  $(\bar{l}_1^1, \bar{l}_1^2, \bar{l}_1^3) = (l_1^1, l_1^2, l_1^3)$ . This along with the deterministic nature of hash tree read write operations means that we must have,

- $(\bar{l}_1^1, \bar{l}_1^2, \bar{l}_1^3) \leftarrow \text{StepR}(\bar{\mathbf{st}}_0)$
- $\{(\bar{b}_1^k, \bar{\Pi}_1^k) := \text{HT.Read}(\text{tree}_0^k, \bar{l}_1^k)\}_{k \in [3]}$
- $(\bar{b}_1^3, \bar{l}_1^3, \bar{\mathbf{st}}_1) := \text{StepW}(\bar{\mathbf{st}}_0, \bar{b}_1^1, \bar{b}_1^2, \bar{b}_1^3)$
- $(\text{tree}_1^3, \bar{\mathbf{rt}}_1^3, \bar{\Pi}_1') := \text{HT.Write}(\text{tree}_0^3, \bar{l}_1^3, \bar{b}_1^3)$

Read and Write Completeness of the hash tree implies

$$\begin{aligned} \text{HT.VerRead}(\text{dk}_1, \bar{r}_0^1, \bar{l}_1^1, \bar{b}_1^1, \bar{\Pi}_1^1) &= 1 \\ \text{HT.VerRead}(\text{dk}_2, \bar{r}_0^2, \bar{l}_1^2, \bar{b}_1^2, \bar{\Pi}_1^2) &= 1 \\ \text{HT.VerRead}(\text{dk}_3, \bar{r}_0^3, \bar{l}_1^3, \bar{b}_1^3, \bar{\Pi}_1^3) &= 1 \\ \text{HT.VerWrite}(\text{dk}_3, \bar{r}_0^3, \bar{l}_1^3, \bar{b}_1^3, \bar{r}_1^3, \bar{\Pi}_1^3) &= 1 \end{aligned}$$

If  $\{b_1^k\}_{k \in [3]} \neq \text{Step}(x, y, P, \text{crs}, 1).\bar{b}$ , then the read soundness assumption of HT implies that

$(\text{HT.VerRead}(\text{dk}, \bar{r}_0^1, \bar{l}_1^k, b_1^k, \Pi_1^k) = 1)_{k \in [3]}$  happens with a negligible probability. Thus, with all but negligible probability we have that  $(C_{\text{index}}, 1) \notin \mathcal{L}^{\text{index}}$  and we are done.

Let us say this is not the case, i.e.,  $\{b_1^k\}_{k \in [3]} = \text{Step}(x, y, P, \text{crs}, 1).\bar{b}$ , then the deterministic nature of the Turing machine write function  $\text{StepW}$  implies that  $\text{st}_1 = \bar{\text{st}}_1$ . Thus, for our assumption to be valid, it must be that  $\{\text{rt}_1^k\}_{k \in [3]} \neq \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{rt}}$ . If  $\text{rt}_1^1 \neq \bar{\text{rt}}_1^1 = \text{rt}_0^1$  or  $\text{rt}_1^2 \neq \bar{\text{rt}}_1^2 = \text{rt}_0^2$ , then the definition of  $\phi$  implies that  $(C_{\text{index}}, 1) \notin \mathcal{L}^{\text{index}}$ . If this is not the case, then the only other possible option is  $\text{rt}_1^3 \neq \bar{\text{rt}}_1^3$ . Now, the write soundness of HT implies that with all but negligible probability,  $\text{HT.VerWrite}(\text{dk}_3, \bar{r}_0^3, \bar{l}_1^3, \bar{b}_1^3, \text{rt}_1^3, \Pi_1^3) \neq 1$  must hold. If this is indeed true then our construction of  $C_{\text{index}}$  in Fig. 3 implies that  $(C_{\text{index}}, 1) \notin \mathcal{L}^{\text{index}}$ .

**Lemma 7.**

$$\Pr[G_{0,c}^A = 1] = \Pr[G_{0,d}^A = 1].$$

*Proof.* Note that by definition,  $\text{h}_1 = \text{st}_1, \text{rt}_1^1, \text{rt}_1^2, \text{rt}_1^3$ . We already have that  $\text{rt}_1^1, \text{rt}_1^2, \text{rt}_1^3 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{rt}}$  and  $\text{st}_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{st}}$ . Thus  $\text{h}_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{h}}$  if and only if  $\text{rt}_1^3 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{rt}} \wedge \text{st}_1 = \text{Step}(x, y, P, \text{crs}, 1).\bar{\text{st}}$ .

**Lemma 8.**

$$\Pr[G_{0,d}^A = 1] \leq \Pr[G_{0,e}^A = 1].$$

*Proof.* The number of conditions for the adversary to win simply decreases from Game  $G_{0,d}$  to Game  $G_{0,e}$ , thus the probability of success must not increase.

A closer observation shows that  $G_{0,e}$  is indeed identical to the case when one puts  $i = 1$  in game  $G_i$ .

Combining these together, we show the base case of the induction to be true. Thus,

$$\Pr[G^A = 1] \leq \Pr[G_1^A = 1] + \text{negl}(\lambda).$$

Assuming that our induction hypothesis holds for some  $j \in [T - 1]$ , we prove that it holds for  $j + 1$  as well. We note that by chain rule, it suffices to show that  $\Pr[G_j^A = 1] \leq \Pr[G_{j+1}^A = 1] + \text{negl}(\lambda)$ . We can show this by a sequence of indistinguishable inner hybrids to transition from Game  $G_j$  to  $G_{j+1}$  which look like the following:

- $G_{j,a}$  Identical to  $G_j$  except the SE hash extraction is done at  $S_{j+1}$  instead of  $S_j$ . Indistinguishability follows from the key indistinguishability of SE hash.
- $G_{j,b}$  Extraction for one of the SE hashes changes from  $I_{S_{j-1}}$  to  $I_{S_{j+1}}$ . However, this does not affect the reduction in any way as extraction at indices  $j-1$  and  $j+1$  are not used by the reduction at any stage.
- $G_{j,c}$  The BARG key generation has a trapdoor at  $j+1$ . This can be done due to key indistinguishability of BARG.
- $G_{j,d}$  The winning condition is changed to: if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \wedge \mathbf{h}_j = \text{Step}(x, y, P, \text{crs}, j) \cdot \mathbf{h}_j \wedge \{b_{j+1}^k\}_{k \in [3]} = \text{Step}(x, y, P, \text{crs}, j+1) \cdot \bar{b}_{j+1} \wedge \text{rt}_{j+1}^3 = \text{Step}(x, y, P, \text{crs}, j+1) \cdot \bar{\text{rt}}_{j+1} \wedge \text{st}_1 = \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\text{st}}$ , return 1.
- $G_{j,e}$  The winning condition is changed to: if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \wedge \mathbf{h}_j = \text{Step}(x, y, P, \text{crs}, j) \cdot \mathbf{h}_j \wedge \{b_{j+1}^k\}_{k \in [3]} = \text{Step}(x, y, P, \text{crs}, j+1) \cdot \bar{b}_{j+1} \wedge \text{rt}_{j+1}^3 = \text{Step}(x, P, \text{crs}, j+1) \cdot \bar{\text{rt}}_{j+1} \wedge \text{st}_1 = \text{Step}(x, y, P, \text{crs}, 1) \cdot \bar{\text{st}} \wedge \mathbf{h}_{j+1} = \text{Step}(x, y, P, \text{crs}, j+1) \cdot \bar{\mathbf{h}}$ , return 1
- $G_{j,f}$  if  $\text{stSNARG.V}(\text{crs}, (x, y), H_P, (c, \Pi)) = 1 \wedge ((x, y), T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}} \wedge \mathbf{h}_{j+1} = \text{Step}(x, y, P, \text{crs}, j+1) \cdot \bar{\mathbf{h}}$ , return 1

The last three steps follow identically as Lemmas 6, 7, 8.

Observe  $G_{j,f}$  is identical to outer Game  $G_{j+1}$  with indices renamed.

Thus, combining the lemmas above, we get

**Lemma 9.** *Assuming extraction correctness of SE, semi-adaptive somewhere soundness of BARG, read and write soundness of HT,*

$$\Pr[G_j^A = 1] \leq \Pr[G_{j+1}^A = 1] + \text{negl}(\lambda).$$

This follows from the combination of previous lemmas where we showed that the winning probability in the sequence of inner hybrids are either negligibly close to each other or increases (from Game  $G_{j,e}$  to Game  $G_{j,f}$ ).

Finally, we will show that the winning probability of  $\mathcal{A}$  is 0 in the final game  $G_T$ .

**Lemma 10.** *Assuming extraction correctness of SE hash,*

$$\Pr[G_T^A = 1] = 0.$$

*Proof.* The extraction correctness of SE ensures that  $\mathbf{h}_T$  was indeed the state committed by the prover. Now,  $\mathbf{h}_T = \bar{\mathbf{h}}_T$  cannot be true since our assumption of  $(x, y, T, P, H_P, \text{crs}) \notin \mathcal{L}_{\mathcal{TM}}$  means that Turing Machine state after  $T$  steps cannot be an accept state. Thus, the adversary's win conditions cannot be simultaneously satisfied.

Note that this step does not require us to resort to BARG soundness. Due to our specific construction of  $\bar{\mathbf{h}}_T$ , all we need ensure is that the state committed by the prover does not correspond to the correct state.

Compiling the lemmas together and using chain rule, it must be true that

$$\Pr[G^A = 1] \leq \text{negl}(\lambda)$$

which is a contradiction to our assumption that the scheme is not sound.



**Lemma 11.** *Assuming  $T = \text{poly}(m, n)$ ,  $T, m, n \leq 2^\lambda$ , the stSNARG protocol in Fig. 2 implies the unconditional existence of a publicly verifiable non interactive succinct delegation scheme sDel as defined above.*

*Proof.* We provide an explicit construction of sDel assuming a semi-trusted SNARG stSNARG. Without loss of generality, we can assume that  $T$  is known a-priory.

- sDel.Setup( $1^\lambda$ ): Run stSNARG.Setup to generate crs.
- sDel.ProgAuth( $1^\lambda, \text{crs}$ ): Generate a program  $P \in \{0, 1\}^m$ , state and run stSNARG.TrustHash(crs,  $P$ ) to get  $H_P$ .
- sDel.I( $1^\lambda, \text{crs}$ ): Generate  $x \in \{0, 1\}^n$ .
- sDel.W(crs,  $P$ , state,  $H_P, x$ ): Generate  $y \in \{0, 1\}$  and run stSNARG.P(crs,  $P, x, y, H_P$ ) to get  $\Pi$ .
- sDel.V(crs,  $x, y, H_P, \Pi$ ): Run stSNARG.V(crs,  $x, y, H_P, \Pi$ ) return V’s output.

Completeness and soundness of sDel follows from the completeness of stSNARG in a straightforward way. Refer to Supplementary material for detailed analysis. The proof size and verifier run time of stSNARG is  $\text{poly}(\lambda, \log T) = \text{poly}(\lambda, \log |P|, \log |x|)$ . Similarly, the prover run time of sDel is also  $\text{poly}(\lambda, |P|, |x|)$ .

## 6 Semi-Trusted Succinct Non-Interactive Argument with Zero Knowledge (ZK-stSNARG)

A publicly verifiable semi-trusted non interactive argument with zero-knowledge scheme ZKstSNARG : (ZKstSNARG.Setup, ZKstSNARG.TrustHash, ZKstSNARG.P, ZKstSNARG.V) is defined as

- ZKstSNARG.Setup( $1^\lambda, 1^T$ ): A randomized setup algorithm which on input security parameter  $\lambda$ , and number of Turing Machine steps  $T$ , outputs crs.
- ZKstSNARG.TrustHash(crs,  $P$ ): A deterministic an honest algorithm which on input crs and a program  $P \in \{0, 1\}^m$  for some  $m < 2^\lambda$ , computes a succinct digest  $H_P$  of  $P$ . It then produces a statistically binding and extractable commitment  $C_P$  of  $H_P$  under randomness  $r_1$ . It then gives out a pair public output POut =  $C_P$  and private output SOut =  $(H_P, r)$ . Here SOut is made available to the prover only.
- ZKstSNARG.P(crs,  $P, x, y, \text{SOut}, \text{POut}$ ): A deterministic prover algorithm which on input the crs,  $P \in \{0, 1\}^m$  for some  $m < 2^\lambda$ ,  $x \in \{0, 1\}^n$  for some  $n < 2^\lambda$ ,  $y \in \{0, 1\}$ , SOut, and POut outputs a proof  $\Pi$ .
- ZKstSNARG.V(crs,  $x, y, \text{POut}, \Pi$ ): A deterministic verification algorithm which on input crs,  $x, y$ , public output POut of stSNARG.TrustHash and proof  $\Pi$ , either accepts(output 1) or rejects(output 0) it.

We define the following language

$$\mathcal{L}_{\mathcal{TM}} := \{(P, x, y, T, \text{POut}, \text{crs}) \mid \exists(H_P, r_1) \text{ such that } \mathcal{TM}(P, x, y) = 1 \wedge (\text{POut}, (H_P, r_1)) = \text{ZKstSNARG.TrustHash}(\text{crs}, P)\}.$$

A ZKstSNARG satisfies the standard completeness, soundness and efficiency properties as stSNARG. It also has an additional property:

**Non Interactive Zero Knowledge.** For all  $(P, x, y, T, \text{POut}, \text{crs}) \in \mathcal{L}_{\mathcal{TM}}$ , there exists a PPT simulator  $\text{Sim} := (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$  such that the distributions of

$$\begin{aligned} (\text{crs}, x, y, \text{POut}, \Pi) | (\text{crs}, \text{aux}) &\leftarrow \text{Sim}_1(1^\lambda, 1^T), \\ (\text{POut}, \text{aux}') &\leftarrow \text{Sim}_2(\text{crs}, \text{aux}), \\ \Pi &\leftarrow \text{Sim}_3(\text{aux}', \text{crs}, (x, y), \text{POut}) \end{aligned}$$

and

$$\begin{aligned} (\text{crs}, x, y, \text{POut}, \Pi) | \text{crs} &\leftarrow \text{ZKstSNARG.Setup}(1^\lambda, 1^T), \\ (\text{POut}, \text{SOut}) &\leftarrow \text{ZKstSNARG.TrustHash}(\text{crs}, P), \\ \Pi &\leftarrow \text{ZKstSNARG.P}(\text{crs}, P, x, y, \text{POut}, \text{SOut}) \end{aligned}$$

are indistinguishable.

To achieve non interactive zero knowledge, we use the following additional primitives, namely (1) a statistically binding extractable commitment scheme  $\text{Com}_{\text{bind}}$  as defined in Sect. 3, and (2) a Non Interactive Zero Knowledge argument  $\text{NIZK} := (\text{NIZK.Gen}, \text{NIZK.P}, \text{NIZK.V})$ .

The protocol in Fig. 5 demonstrates the extension of stSNARG to achieve Zero-Knowledge. The CRS in Fig. 5 contains a statistically binding commitment to 0. This lets us extend  $\mathcal{L}_{\mathcal{TM}}$  to the language,

$$\begin{aligned} \mathcal{L}_{\text{hyb}} := &\left\{ (P, x, y, T, C_P, \text{crs}) \mid \exists (H_P, r_1) \text{ such that } \mathcal{TM}(P, x, y) = 1 \right. \\ &\quad \wedge (C_P, (H_P, r_1)) = \text{ZKstSNARG.TrustHash}(\text{crs}, P) \\ &\quad \left. \vee (\exists r \text{ such that } \text{crs} \text{ contains a commitment to } 1 \text{ under randomness } r) \right\} \end{aligned}$$

such that any witness to  $\mathcal{L}_{\mathcal{TM}}$  is vacuously a witness to  $\mathcal{L}_{\text{hyb}}$  due to binding property of the commitment. We use NIZK for the following NP language:

$$\begin{aligned} \mathcal{L} := &\left\{ (c.\text{com}, \Pi.\text{com}, (\text{crs}, x, y, T), C_P) \mid \exists r_1, r_2, r_3, r_4, c, \Pi, H_P \text{ such that} \right. \\ &\quad \left( C_P = \text{Com.C}(\text{Com}_{\text{bind}}.\text{Key}_1, H_P; r_1) \wedge c.\text{com} = \text{Com.C}(\text{Com}_{\text{bind}}.\text{Key}_2, c; r_2) \right. \\ &\quad \left. \wedge \Pi.\text{com} = \text{Com.C}(\text{Com}_{\text{bind}}.\text{Key}_3, \Pi; r_3) \wedge \text{stSNARG.V}(\text{crs}, ((x, y), T, H_P), (c, \Pi)) = 1 \right) \\ &\quad \left. \vee \text{crs contains Com.C}(\text{Com}_{\text{bind}}.\text{Key}_4, 1; r_4) \right\} \end{aligned}$$

Also, note that in this construction, the underlying stSNARG is built for the **index circuit**  $C'_{\text{index}}$  which is identical to  $C_{\text{index}}$  except that  $H_P$  is a part of the input and not hard-coded in the circuit as it is not known to the verifier.

**Theorem 4.** *Assuming the existence of semi-trusted SNARGs and Extractable Statistically Binding Commitment Schemes, and NIZK as described in Sects. 3 and 5, Fig. 5 is a publicly verifiable non-interactive semi-trusted SNARG with zero knowledge such that CRS size, proof size and verifier time are  $\text{poly}(\lambda, \log T)$  and prover run time is  $\text{poly}(\lambda, T)$ .*

**Protocol 2 (stSNARG with Zero-Knowledge)**

- $\text{ZKstSNARG.Setup}(1^\lambda, T)$  :
  - $\text{crs}_1 \leftarrow \text{stSNARG.Setup}(1^\lambda, 1^T)$
  - $\text{Com}_{\text{bind}}.\text{Key}_1 \leftarrow \text{Com.Gen}(1^\lambda)$
  - $\text{Com}_{\text{bind}}.\text{Key}_2 \leftarrow \text{Com.Gen}(1^\lambda)$
  - $\text{Com}_{\text{bind}}.\text{Key}_3 \leftarrow \text{Com.Gen}(1^\lambda)$
  - $\text{Com}_{\text{bind}}.\text{Key}_4 \leftarrow \text{Com.Gen}(1^\lambda)$
  - $r_4 \leftarrow \text{Com.C}(\{0, 1\}^\lambda, z \leftarrow \text{Com.C}(\text{Com.Key}_4, 0; r_4))$
  - $\text{NIZK.crs} \leftarrow \text{NIZK.Gen}(1^\lambda)$
  - *return*  $(\text{crs}_1, \text{Com}_{\text{bind}}.\text{Key}_1, \text{Com}_{\text{bind}}.\text{Key}_2, \text{Com}_{\text{bind}}.\text{Key}_3, z, \text{NIZK.crs})$ .
- $\text{ZKstSNARG.TrustHash}(\text{crs}, P)$ 
  - $H_P \leftarrow \text{stSNARG.TrustHash}(\text{crs}, P)$
  - $r_1 \leftarrow \text{Com.C}(\{0, 1\}^\lambda, C_P \leftarrow \text{Com.C}(\text{Com}_{\text{bind}}.\text{Key}_1, H_P; r_1))$  *return*  $(\text{SOut} := (P, r_1), \text{POut} := C_P)$ .
- $\text{ZKstSNARG.P}(\text{crs}, x, y, \text{SOut}, \text{POut})$  :
  - $(c, \Pi) \leftarrow \text{stSNARG.P}(\text{crs}, x, y, H_P)$
  - $r_2 \leftarrow \text{Com.C}(\{0, 1\}^\lambda, c.\text{com} \leftarrow \text{Com.C}(\text{Com}_{\text{bind}}.\text{Key}_2, c; r_2))$
  - $r_3 \leftarrow \text{Com.C}(\{0, 1\}^\lambda, \Pi.\text{com} \leftarrow \text{Com.C}(\text{Com}_{\text{bind}}.\text{Key}_3, \Pi; r_3))$
  - $\text{NIZK}.\Pi \leftarrow \text{NIZK.Prove}(\text{NIZK.crs}, (c.\text{com}, \Pi.\text{com}, (\text{crs}, x, y, T), C_P), ((H_P, r_1), (c, r_2), (\Pi, r_3), \perp))$
  - *return*  $(c.\text{com}, \Pi.\text{com}, \text{NIZK}.\Pi)$ .
- $\text{ZKstSNARG.V}(\text{crs}, (x, y), \text{POut} = C_P, c.\text{com}, \Pi.\text{com}, \text{NIZK}.\Pi)$  :
  - *return* 1 if and only if  $\text{NIZK.V}(\text{NIZK.crs}, (c.\text{com}, \Pi.\text{com}, (\text{crs}, x, y, T), C_P), \text{NIZK}.\Pi) = 1$ .

**Fig. 5.** Semi-Trusted Universal Turing Machine Delegation with Non Interactive Zero-Knowledge

*Completeness and Efficiency.* Completeness follows from the completeness of the underlying stSNARG, NIZK and the binding property of the commitment. Similarly succinctness follows from the efficiency of stSNARG, NIZK, and the binding commitment  $\text{Com}_{\text{bind}}$ .

*Soundness.* The soundness following by a straightforward reduction using the CRS indistinguishability and Statistical Binding of  $\text{Com}_{\text{bind}}$ , and the soundness of the underlying stSNARG. We can construct an adversary that breaks the soundness of the underlying stSNARG using the following steps:

- 1 Change the keys for  $\text{Com}_{\text{bind}}$  to be generated by TGen. This can be done due to CRS indistinguishability.

- 2 The reduction can now extract the committed proof from  $c.com$  and  $\Pi.com$ . This is because the reduction has access to the trapdoor commitment key.
- 3 The  $stSNARG$  can now output the extracted proof. The extraction correctness of  $Com_{bind}$  ensures that if  $ZKstSNARG$  is not sound, then this adversary breaks the soundness of  $stSNARG$ .

A formal analysis is presented in the Supplementary Material.

*Zero-Knowledge.*

$zk - stSNARG$  Simulator  $NIZK.Sim := (Sim_1, Sim_2, Sim_3)$

- $Sim_1(1^\lambda, 1^T)$  :
  1.  $SE.K_{even} \leftarrow SE.Gen(1^\lambda, 1^{M_{\lambda,T}}, 1^{L_\lambda})$
  2.  $SE.K_{odd} \leftarrow SE.Gen(1^\lambda, 1^M, 1^L)$
  3.  $BARG.crs \leftarrow BARG.Gen(1^\lambda, 1^{T+1}, 1^{|C_{index}|})$
  4.  $dk \leftarrow HT.Gen(1^\lambda)$
  5.  $Com_{bind}.Key_1 \leftarrow Com.Gen(1^\lambda)$
  6.  $Com_{bind}.Key_2 \leftarrow Com.Gen(1^\lambda)$
  7.  $Com_{bind}.Key_3 \leftarrow Com.Gen(1^\lambda)$
  8.  $Com_{bind}.Key_4 \leftarrow Com.Gen(1^\lambda), r_4 \leftarrow \mathbb{N}, z \leftarrow Com.C(Com.Key_4, 1; r_4)$
  9.  $NIZK.crs \leftarrow NIZK.Gen(1^\lambda)$
  10. return  $crs := (SE.K_{even}, SE.K_{odd}, BARG.crs, dk, Com_{bind}.Key_1, Com_{bind}.Key_2, Com_{bind}.Key_3, z, NIZK.crs)$  and  $aux := r_4$
- $Sim_2(crs, aux)$  :
  1.  $r_1 \leftarrow \mathbb{N}, C_P \leftarrow Com.C(Com_{bind}.Key_1, 0; r_1)$  return  $POut := C_P$ .
  2. return  $(POut, aux' := aux)$
- $Sim_3(crs, aux', (x, y), POut := C_P)$  :
  1.  $r_2 \leftarrow \mathbb{N}, c.com \leftarrow Com.C(Com_{bind}.Key_2, 0; r_2)$
  2. Generate a dummy proof  $\hat{\Pi}$
  3.  $r_3 \leftarrow \mathbb{N}, \Pi.com \leftarrow Com.C(Com_{bind}.Key_3, 0; r_3)$
  4.  $NIZK.\Pi \leftarrow NIZK.Prove(NIZK.crs, (c.com, \Pi.com, (crs, x, y, T), C_P), (\perp, \perp, \perp, aux'))$
  5. return  $(c.com, \Pi.com, NIZK.\Pi)$ .

The proof of zero-knowledge follows from a sequence of hybrids.

- We define a game  $G'$  which is identical to  $G_0$  except that  $crs$  has a commitment of 1 instead of 0. Note that an honest prover does not make use of this section of the  $crs$  in its proof. Consider  $hyb'$  as the output distribution of intermediate  $G'$ . All other algorithms in  $G'$  remains identical as  $G_0$ .  $hyb_0$  must be indistinguishable from  $hyb'$ , otherwise we can construct an efficient adversary that breaks the computational hiding property of  $Com_{bind}$ .
- The hybrid game  $G''$  with output distribution  $hyb''$  works like  $G'$  except  $stSNARG.P$  computes  $(c.com, NIZK.\Pi)$  honestly and then ignores  $c.com$  and outputs  $(c_1, NIZK.\Pi)$  where  $c_1$  is the statistical binding commitment to the 0 string using  $Com_{bind}$ . The indistinguishability of  $hyb'$  and  $hyb''$  follows from the computational hiding property of  $Com_{bind}$ .

- We now define another hybrid game  $G'''$  where everything remains identical as  $G''$  but the NIZK proof  $\text{NIZK.P}$  proves that  $\text{crs}$  has a commitment of 1 using randomness  $r$  as a witness. This is indeed a valid witness for the same language  $\mathcal{L}_{\text{hyb}}^*$ . Observe that  $G''$  and  $G'''$  have identical CRS. However,  $\text{NIZK.P}$  in each case uses different witnesses, namely  $r$  and  $((c, r_{\text{com}_2}), \Pi)$  respectively. Thus, the Witness Indistinguishability of NIZK implies indistinguishability of  $G''$  and  $G'''$ .
- In the next hybrid  $G''''$ , trusted commitment generator is replaced by  $\text{Sim}_2$  which on input  $\text{crs}$  simply outputs a hiding commitment to the 0 string. Note that the output of  $\text{Sim}_2$  is not used anywhere else in the proof and its output is computationally indistinguishable from the public output of  $\text{ZKstSNARG.TrustHash}(\text{crs}, P)$  because of the hiding property of commitment scheme.
- In the final game  $G_1$ ,  $\text{Sim}_1$  uses the same  $\text{crs}$  as the previous hybrid.  $\text{Sim}_3$  ignores all operations performed by the prover and only outputs  $c_1$  which is the statistical binding commitment to the 0 string using  $\text{Com}_{\text{bind}}$  and sends a NIZK proof as  $G''''$ . The output distributions of  $G''''$  and  $G_1$  are indeed identical as the output of  $\text{Sim}_3$  solely depends on the output of  $\text{Sim}_1, \text{Sim}_2$  and the commitment of the 0 string  $c_1$ .

Combining all the hybrids, we prove that  $G_0$  and  $G_1$  have output distributions which are computationally indistinguishable.

*Public Verifiable Non Interactive Succinct Delegation with Zero Knowledge.* A direct extension of Lemma 11 gives us the following corollary,

**Corollary 2.** *Assuming  $T = \text{poly}(m, n)$ ,  $T, m, n \leq 2^\lambda$ , the ZKstSNARG protocol in Fig. 5 implies the unconditional existence of a publicly verifiable non interactive succinct delegation scheme with zero knowledge.*

The zero knowledge simulator for the delegation scheme  $\text{zk-sDel.Sim} := (\text{zk-sDel.Sim}_1, \text{zk-sDel.Sim}_2)$  can simply run the  $\text{stSNARG ZK-simulator}$ . More specifically,  $\text{zk-sDel.Sim}_1$  and  $\text{zk-sDel.Sim}_2$  call  $\text{zk-stSNARG.Sim}_1$  and  $\text{zk-stSNARG.Sim}_2$  respectively above. The proof follows in a straightforward manner, hence we skip the details.

**Acknowledgements.** This research was supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024.

## References

1. Badrinarayanan, S., Kalai, Y.T., Khurana, D., Sahai, A., Wichs, D.: Succinct delegation for low-space non-deterministic computation. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) 50th ACM STOC, pp. 709–721. ACM Press (2018)

2. Bitansky, N., et al.: The hunting of the snark. *J. Cryptol.* **30**(4), 989–1066 (2017)
3. 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 (2013)
4. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (Mar (2013)). [https://doi.org/10.1007/978-3-642-36594-2\\_18](https://doi.org/10.1007/978-3-642-36594-2_18)
5. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) 50th ACM STOC, pp. 671–684. ACM Press (2018)
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 (2017)
7. Brakerski, Z., Kalai, Y.: Witness indistinguishability for any single-round argument with applications to access control. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 97–123. Springer, Heidelberg (May (2020)). [https://doi.org/10.1007/978-3-030-45388-6\\_4](https://doi.org/10.1007/978-3-030-45388-6_4)
8. Canetti, R., et al.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC, pp. 1082–1090. ACM Press (2019)
9. Choudhuri, A.R., Jain, A., Jin, Z.: Non-interactive batch arguments for NP from standard assumptions. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 394–423. Springer, Heidelberg, Virtual Event (Aug (2021)). [https://doi.org/10.1007/978-3-030-84259-8\\_14](https://doi.org/10.1007/978-3-030-84259-8_14)
10. Choudhuri, A.R., Jain, A., Jin, Z.: SNARGs for  $\mathcal{P}$  from *lwe*. Cryptology ePrint Archive (2021)
11. Chung, K.M., Kalai, Y., Vadhan, S.P.: Improved delegation of computation using fully homomorphic encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (Aug (2010)). [https://doi.org/10.1007/978-3-642-14623-7\\_26](https://doi.org/10.1007/978-3-642-14623-7_26)
12. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 90–112 (2012)
13. 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 (Mar (2012)). [https://doi.org/10.1007/978-3-642-28914-9\\_4](https://doi.org/10.1007/978-3-642-28914-9_4)
14. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string. In: Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science, pp. 308–317. IEEE (1990)
15. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (Aug (2010)). [https://doi.org/10.1007/978-3-642-14623-7\\_25](https://doi.org/10.1007/978-3-642-14623-7_25)
16. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EURO-CRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May (2013)). [https://doi.org/10.1007/978-3-642-38348-9\\_37](https://doi.org/10.1007/978-3-642-38348-9_37)
17. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. *J. ACM (JACM)* **62**(4), 1–64 (2015)

18. 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](https://doi.org/10.1007/978-3-642-17373-8_19)
19. Holmgren, J., Lombardi, A., Rothblum, R.D.: Fiat-shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 750–760 (2021)
20. 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 (2015)
21. Kalai, Y., Paneth, O., Yang, L.: On publicly verifiable delegation from standard assumptions. Cryptology ePrint Archive (2018)
22. Kalai, Y.T., Paneth, O.: Delegating RAM computations. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 91–118. Springer, Heidelberg (Oct / Nov (2016)). [https://doi.org/10.1007/978-3-662-53644-5\\_4](https://doi.org/10.1007/978-3-662-53644-5_4)
23. 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 (2019)
24. Kalai, Y.T., Raz, R., Rothblum, R.D.: Delegation for bounded space. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 565–574. ACM Press (2013)
25. Kalai, Y.T., Raz, R., Rothblum, R.D.: How to delegate computations: the power of no-signaling proofs. In: Shmoys, D.B. (ed.) 46th ACM STOC, pp. 485–494. ACM Press (2014)
26. Kalai, Y.T., Vaikuntanathan, V., Zhang, R.Y.: Somewhere statistical soundness, post-quantum security, and SNARGs. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13042, pp. 330–368. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-90459-3\\_12](https://doi.org/10.1007/978-3-030-90459-3_12)
27. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing, pp. 723–732 (1992)
28. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (Mar (2012)). [https://doi.org/10.1007/978-3-642-28914-9\\_10](https://doi.org/10.1007/978-3-642-28914-9_10)
29. Lombardi, A., Schaeffer, L.: A note on key agreement and non-interactive commitments. Cryptology ePrint Archive (2019)
30. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (Aug (1988)). [https://doi.org/10.1007/3-540-48184-2\\_32](https://doi.org/10.1007/3-540-48184-2_32)
31. Micali, S.: Computationally sound proofs. SIAM J. Comput. **30**(4), 1253–1298 (2000)
32. Paneth, O., Rothblum, G.N.: On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 283–315. Springer, Heidelberg (Nov (2017)). [https://doi.org/10.1007/978-3-319-70503-3\\_9](https://doi.org/10.1007/978-3-319-70503-3_9)
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 (2013)
34. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (Mar (2012)). [https://doi.org/10.1007/978-3-642-28914-9\\_24](https://doi.org/10.1007/978-3-642-28914-9_24)

35. 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, Heidelberg (Aug (2019)). [https://doi.org/10.1007/978-3-030-26948-7\\_4](https://doi.org/10.1007/978-3-030-26948-7_4)
36. Reingold, O., Rothblum, G.N., Rothblum, R.D.: Constant-round interactive proofs for delegating computation. SIAM J. Comput. **50**(3), STOC16-255 (2019)
37. Setty, S., Vu, V., Panpalia, N., Braun, B., Blumberg, A.J., Walfish, M.: Taking *{Proof – Based}* verified computation a few steps closer to practicality. In: 21st USENIX Security Symposium (USENIX Security 12), pp. 253–268 (2012)
38. Setty, S.T., McPherson, R., Blumberg, A.J., Walfish, M.: Making argument systems for outsourced computation practical (sometimes). In: NDSS, vol. 1, p. 17 (2012)
39. Thaler, J., Roberts, M., Mitzenmacher, M., Pfister, H.: *{Verifiable}* computation with massively parallel interactive proofs. In: 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 12) (2012)





# Laconic Function Evaluation for Turing Machines

Nico Döttling<sup>1</sup>, Phillip Gajland<sup>2,3(✉)</sup>, and Giulio Malavolta<sup>2</sup>

<sup>1</sup> CISA Helmholtz Center for Information Security, Saarbrücken, Germany  
doettling@cispa.de

<sup>2</sup> Max Planck Institute for Security and Privacy, Bochum, Germany  
{phillip.gajland,giulio.malavolta}@mpi-sp.org

<sup>3</sup> Ruhr-University Bochum, Bochum, Germany

**Abstract.** Laconic function evaluation (LFE) allows Alice to compress a large circuit  $\mathbf{C}$  into a small digest  $\mathbf{d}$ . Given Alice's digest, Bob can encrypt some input  $x$  under  $\mathbf{d}$  in a way that enables Alice to recover  $\mathbf{C}(x)$ , without learning anything beyond that. The scheme is said to be *laconic* if the size of  $\mathbf{d}$ , the runtime of the encryption algorithm, and the size of the ciphertext are all sublinear in the size of  $\mathbf{C}$ .

Until now, all known LFE constructions have ciphertexts whose size depends on the *depth* of the circuit  $\mathbf{C}$ , akin to the limitation of *levelled* homomorphic encryption. In this work we close this gap and present the first LFE scheme (for Turing machines) with asymptotically optimal parameters. Our scheme assumes the existence of indistinguishability obfuscation and somewhere statistically binding hash functions. As further contributions, we show how our scheme enables a wide range of new applications, including two previously unknown constructions:

- Non-interactive zero-knowledge (NIZK) proofs with optimal prover complexity.
- Witness encryption and attribute-based encryption (ABE) for Turing machines from falsifiable assumptions.

---

Nico Döttling—Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Phillip Gajland—Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and the European Union (ERC AdG REWORC - 101054911).

Giulio Malavolta—Funded by the German Federal Ministry of Education and Research BMBF (grant 16K15K042, project 6GEM) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038.

© International Association for Cryptologic Research 2023

A. Boldyreva and V. Kolesnikov (Eds.): PKC 2023, LNCS 13941, pp. 606–634, 2023.

[https://doi.org/10.1007/978-3-031-31371-4\\_21](https://doi.org/10.1007/978-3-031-31371-4_21)

# 1 Introduction

Laconic function evaluation (LFE) is a cryptographic primitive recently introduced by Quach, Wee, and Wichs [FOCS’18]. Using LFE, Alice can compress a large circuit  $\mathbf{C}$  into a small digest  $\mathbf{d}$ . Given Alice’s digest, Bob can encrypt some input  $x$  under  $\mathbf{d}$  in a way that enables Alice to recover  $\mathbf{C}(x)$  without learning anything about Bob’s input. The scheme is said to be *laconic* if the size of the digest  $\mathbf{d}$ , the runtime of the encryption algorithm  $\text{LFE.Enc}$ , and the size of the ciphertext  $\mathbf{c}$  are all sublinear in the size of  $\mathbf{C}$ .

LFE is particularly interesting in the context of two-party and multi-party computation (2PC, MPC), since it enables the construction of protocols with novel properties. As an example, LFE enables a “Bob-optimised” two-round 2PC protocol in which Alice does all the work, while Bob’s computation and communication are smaller than both the function being evaluated and Alice’s input. However, for all known LFE constructions [QWW18, AR21, NRS21], the runtime of the encryption procedure and the size of Bob’s ciphertext depend on the *depth* of the circuit being evaluated by Alice. This is a severe limitation which restricts the applicability of this primitive to “shallow” circuits. In some sense, this mirrors the efficiency gap between *levelled* and *fully* homomorphic encryption. This leaves us with the following open problem (also stated in [QWW18]):

*Is it possible to construct LFE where Bob’s work is independent of the circuit size?*

## 1.1 Our Results

We answer this question in the affirmative and our main result is the construction of an asymptotically optimal LFE scheme assuming indistinguishability obfuscation [BGI+01] and somewhere statistically binding (SSB) hash functions [HW15]. Our construction enables the computation of any Turing machine  $\mathbf{M}$  and, unlike all prior constructions [QWW18] [AR21] [NRS21], removes the dependency on the depth of the circuit (the runtime of the Turing machine in our case). In the standard simulation-security setting, we obtain the following result.

**Theorem 1 (Informal).** *Assuming indistinguishability obfuscation for circuits and somewhere statistically binding hash functions, there exists a simulation secure LFE scheme with the following parameters:*

- The size of the digest  $\mathbf{d}$  is  $\text{poly}(\lambda)$ .
- The runtime of the encryption procedure is  $\mathcal{O}(|x| + |\mathbf{M}(x)|) \cdot \text{poly}(\lambda)$ .
- The size of the ciphertext  $\mathbf{c}$  is  $\mathcal{O}(|x| + |\mathbf{M}(x)|) \cdot \text{poly}(\lambda)$ .

If we relax the security to an indistinguishability-based notion, we can further improve the parameters by removing the dependency on the size of the output.

**Theorem 2 (Informal).** *Assuming indistinguishability obfuscation for circuits and somewhere statistically binding hash functions, there exists a LFE scheme with ciphertext indistinguishability and the following parameters:*

- The size of the digest  $d$  is  $\text{poly}(\lambda)$ .
- The runtime of the encryption procedure is  $\mathcal{O}(|x|) \cdot \text{poly}(\lambda)$ .
- The size of the ciphertext  $c$  is  $\mathcal{O}(|x|) \cdot \text{poly}(\lambda)$ .

As for the underlying assumptions, SSB hash functions [OPWW15] can be constructed from a variety of standard assumptions (e.g. LWE or DDH), whereas indistinguishability obfuscation is a less understood primitive and currently the subject of a large body of research. Numerous recent works [BDGM20, GP20, JLS20, WW21] show provably-secure constructions of indistinguishability obfuscation for circuits under simple assumptions, some of which are regarded as well-founded.

We briefly describe some additional contributions which show how our construction enables a wide range of new results in cryptography.

**(1) Witness Encryption for Turing Machines:** We construct the first witness encryption where the size of the ciphertext depends only on the size of the witness and the security parameter (but not on the NP relation  $\mathcal{R}$ ). Furthermore, the decryption runtime is only proportional to the runtime of the Turing machine computing  $\mathcal{R}$ , rather than its circuit representation. This implies the first ABE for Turing machines [GKP+13] from falsifiable assumptions. Prior to our work, Goldwasser et al. [GKP+13] constructed the same primitive from *extractable* witness encryption,<sup>1</sup> which is a considerably stronger and non-falsifiable assumption, whose validity has often been called into question [GGHW14, BP15, BSW16].

**(2) NIZKs with Optimal Prover Complexity:** By applying a known transformation [KNYY19], we construct the first *prover-optimal* NIZK proof system, where the prover’s computational complexity depends only on the size of the witness and on the security parameter (and is otherwise independent of the size of the NP relation).

**(3) MPC Compiler:** By applying the transformation described in [QWW18] we obtain a compiler for multi-party computation (MPC) that reduces the communication complexity to be independent of the circuit size, *without introducing additional rounds of interaction*.

## 1.2 Technical Overview

Following is a brief overview of the techniques developed in this work. Before delving into our approach, we briefly discuss why trivial solutions fall short in constructing LFE.

<sup>1</sup> We should also mention a recent work of Ananth et al. [AFS19], which constructs ABE for RAM programs from LWE, although it achieves only a weaker form of efficiency where the public parameters and the ciphertexts grow with the runtime of the RAM program.

**Why Trivial Solutions Fail.** An astute reader may wonder why this is still a challenging problem, given  $\text{iO}$  for circuits. One plausible approach to constructing LFE via this route would be to place the hash of the circuit  $\mathbf{d} := H(\mathbf{C})$  in the common reference string. Bob could then obfuscate and send Alice the following universal circuit

$$\mathcal{U}(\mathbf{C}') : \text{if } \mathbf{d} \stackrel{?}{=} H(\mathbf{C}') \text{ return } \mathbf{C}'(x).$$

Intuitively, Alice should only be able to run the obfuscated circuit on  $\mathbf{C}$  unless she is able to find a collision for  $H$ . Unfortunately, this approach has two major flaws:

- (1) Efficiency: The construction is *not laconic* since both the runtime of Bob and the size of the ciphertext depend on the size of  $\mathbf{C}$ . Even recent constructions of  $\text{iO}$  for Turing machines [AJS17] suffer from the drawback that the size of the obfuscated Turing machine depends on the maximum input size. An exception is the recent work of [BFK+19] which, however, requires a large shared random string or a random oracle. At present, constructing  $\text{iO}$  without input-size dependence remains an open problem.
- (2) Provable Security: The above informal argument assumes the strong notion of virtual-blackbox obfuscation, which is known to be impossible [BGI+01]. Constructing a provably secure scheme requires a significant modification of the template in order to be able to leverage the weak *indistinguishability* security of  $\text{iO}$ .

Even if  $\text{iO}$  for Turing machines does not appear to be sufficient to construct LFE, it turns out that other techniques from the area [KLW15, CCHR15, CH16, CCC+16, ACC+16, GS18] will help us in building a provably-secure scheme, as we explain in the following.

To understand the challenge in more detail, it is useful to compare the notion of LFE with succinct randomized encodings (SRE) [BGL+15]: SRE allows one to encode an input  $x$  with respect to a public Turing machine  $M$  in such a way that nothing is revealed beyond  $M(x)$ . However, the *runtime of the encoding algorithm and the size of the encoding* depend on the size of  $M$ , whereas in LFE Bob’s ciphertext crucially only depends on the size of his input (and the security parameter). Furthermore, SRE do not allow Alice to privately hash her circuit/Turing machine.

**Our Approach and Differences to [GS18].** Readers familiar with the work of [GS18] may wonder why their results cannot be used “off the shelf” as follows. Alice computes the digest of her encrypted input along with the circuit  $\mathbf{C}$ . Then she sends the resulting hash to Bob, who computes a succinct randomised encoding as specified in [GS18], except using Alice’s digest. Then Alice can just load  $\mathbf{C}$  into the memory of the Turing machine  $M$ , thus allowing us to use the result of [GS18] off-the-shelf. Unfortunately, this solution does not work, as [GS18] states that the hash is binding only for the *non- $\perp$*  locations of the database (whose length is denoted by the parameter  $n$ ). This raises the question whether adding  $\mathbf{C}$  to the database should result in a hash which is “binding for  $\mathbf{C}$ ”. - If yes, namely the hash is binding for  $\mathbf{C}$ , then the parameter

$n$  will grow with the size of  $|\mathbf{C}|$ . In this case, the complexity of hash update (and consequently the runtime of Bob) will be  $\text{poly}(n, \lambda, \log M)$ . Here,  $M$  denotes the size of the database. Since  $n \geq |\mathbf{C}|$ , this means that the runtime of Bob would depend on the size of  $\mathbf{C}$ , which nullifies Bob's efficiency. - If no, namely the hash is not binding for  $\mathbf{C}$ , then we do not see a direct way to prove the security of the construction, since we cannot rule out that Alice knows a different circuit  $\mathbf{C}'$  that collides with  $\mathbf{C}$ , thus breaking the security of the encryption. Note that the hash is compressing, so collisions always exist even if the hash is computed honestly. The key observation here is that the size of [GS18]'s hash doesn't depend on the size of the unassigned ( $= \perp$ ) locations, but *does* depend on the number of specified locations ( $= n$ ). As such loading the circuit  $\mathbf{C}$  into memory, would increase the number of specified locations.

Our construction builds on the techniques introduced in [GS18], and requires us to modify the construction in a non-blackbox manner, in order to constrain Alice to execute the Turing machine  $M$  on Bob's input while at the same time making Bob's runtime independent of it. To gain some intuition on the approach, we consider the simplified setting in which both parties know a public Turing machine  $M$ , where the transition function is denoted by  $\mathbf{C}_M$  and Bob holds an input  $x$ . Later in this overview, we show that this template can be lifted to the more generic setting where Alice evaluates a *private* Turing machine by letting  $M$  be a universal Turing machine with an additional input. To establish some notation, consider the insecure protocol where Bob sends his input  $x$  in plain: Alice can evaluate  $M$  by maintaining a database  $D$  that encodes  $x$  and the current state of the memory of  $M$ . Each operation of  $\mathbf{C}_M$  consists of reading the current state, one bit from Alice, and one from Bob.

**Garbled Circuits.** One possible way to secure this approach is to use Yao's garbled circuits [Yao82, Yao86], that allow for the secure computation of a circuit  $\mathbf{C}$  by creating a *garbled* version  $\tilde{\mathbf{C}}$  and encoding the input  $x = (x_1, \dots, x_n)$  as a set of labels  $(\text{lbs}_1, \dots, \text{lbs}_n)$ . Security is guaranteed as long as a *single* input encoding is revealed to the evaluator. If we were to garble the step circuit  $\mathbf{C}_M$ , we immediately run into two problems: (1) From an efficiency perspective, Bob would need to garble one circuit for each step of the computation, which would be more expensive than just evaluating  $M$  locally. (2) With regards to functionality, the evaluator needs to receive the labels corresponding to an input encoding. This corresponds to a particular set of locations in  $D$  (depending on which bits  $\mathbf{C}_M$  needs to read). The difficulty here stems from the fact that the state of  $D$  evolves over the course of the computation, as it includes the memory tape of the Turing machines. Thus, we would need a way to *dynamically* select labels depending on the intermediate state of  $D$ . Fortunately, (1) can be solved using  $\text{iO}$ : Instead of garbling all step circuits explicitly, Bob sends an obfuscated circuit that, given an index  $i$ , returns the  $i^{\text{th}}$  garbled step circuit. The remainder of this overview is devoted to solving (2).

**Updatable Laconic Oblivious Transfer.** Before explaining our solution, we recall the notion of updatable laconic oblivious transfer (ULOT) [CDG+17].

With an ULOT protocol, a large database  $D$  can be hashed to a small digest  $\mathbf{d}$  offering the sender two operations.

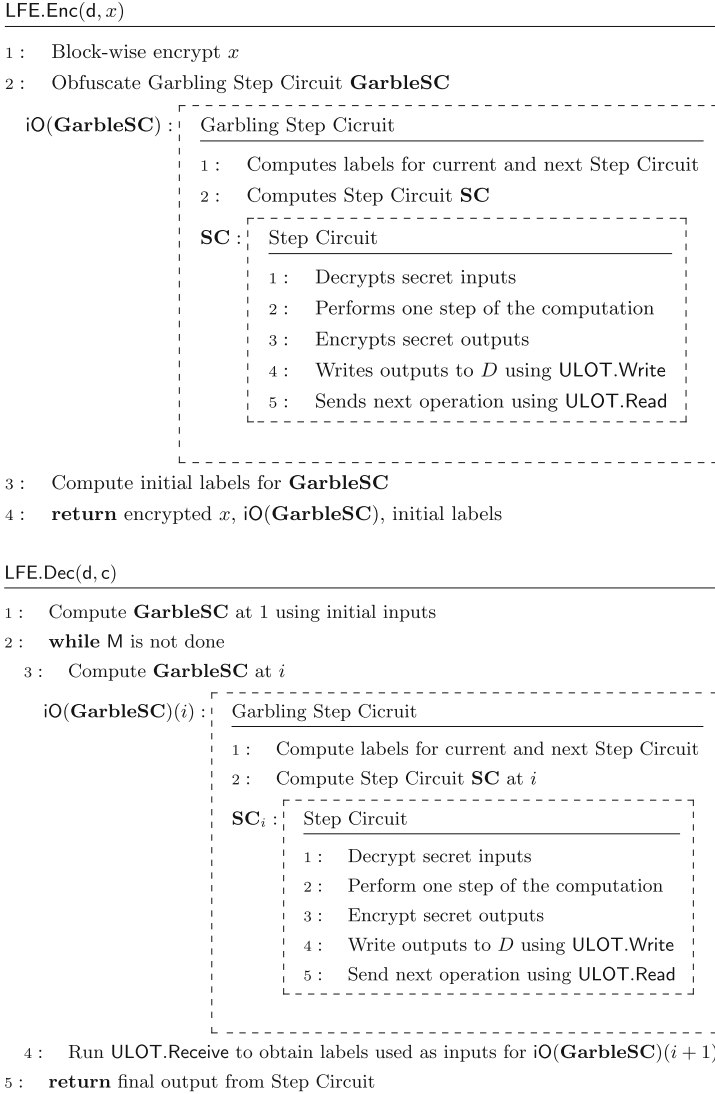
**Read:** Given a pair of messages  $(m_0, m_1)$  and an index  $i$ , the sender can compute a ciphertext  $c$  such that the receiver (knowing  $D$  and  $\mathbf{d}$ ) can recover  $m_{D[i]}$ , where  $D[i]$  is the value of the bit at the  $i$ th location of  $D$ .

**Write:** Given  $|\mathbf{d}|$ -many pairs of messages  $\{m_{0,i}, m_{1,i}\}_{i \in [|\mathbf{d}|]}$ , a bit  $b$ , and index  $i$ , the sender can compute a ciphertext  $c$  such that the receiver (knowing  $D$  and  $\mathbf{d}$ ) can recover  $(m_{D'_1,1}, \dots, m_{D'_{|\mathbf{d}|},|\mathbf{d}|})$ . Here,  $\mathbf{d}'$  is the hash of  $D'$ , the database  $D$  updated by writing  $b$  at index  $i$ .

Equipped with this functionality, we can now devise a mechanism to provide the evaluator with the appropriate input encodings. Bob compresses his input  $x$ , using the hashing procedure of the ULOT scheme and sends it to Alice, who will act as the evaluator. At each step of the computation, Alice is provided with the labels corresponding to the database locations needed by the current step circuit. She then uses these labels to evaluate the garbled step circuit, which performs the computation step and computes a ULOT ciphertext containing the pairs of labels for the next step of the computation. In the next step, Alice will be able to retrieve the set corresponding to the locations of the updated database, by running the receive algorithm of the ULOT. These include an encoding of the updated hash of  $D$ , as a result of the write operation of the step circuit.

**Piecing It Together.** Now only two problems remain. First, the state of  $D$  is given in clear to Alice, meaning the intermediate values of the computation are leaked. This is solved by adding a layer of symmetric encryption to the memory of the Turing machine. To ensure the correctness of the computation, we remove this layer before feeding the input into  $\mathbf{C}_M$ . The output is then re-encrypted using a new key that is only available in the next step circuit. As this happens within the garbled circuit, security is preserved. We can now lift the construction to the setting where Alice's  $M$  is not known to Bob. This is done by including an additional ULOT digest of the description of the Turing machine, which allows the step circuit to read the description (via ULOT read) and determines the next operation of the computation. Given the above procedure, the database lookup algorithm can be naturally extended to the case of an additional tape, encoding the machine's instructions. To ensure that the random coins used in the garbled circuits are *consistent* across different computation steps, we use a (puncturable) PRF to sample the labels.

**The Final Scheme.** We provide some intuition for the encryption and decryption procedures in Fig. 1. For the encryption procedure, Bob starts by obfuscating the Garbling Step Circuit and computing the first set of labels that will be needed to evaluate the garbled circuit. These are then sent along with his encrypted input to Alice. For the decryption procedure, Alice evaluates the garbled circuit using the first set of labels sent by Bob. The output from the Step Circuit is then used for receiving the updatable laconic oblivious transfer. This is repeated for all steps of the computation until the final output is returned by the decryption procedure.



**Fig. 1.** High level overview of the encryption and decryption procedures.

**Security Proof.** Next, we provide some intuition about the security argument. To prove the security of our construction we use a similar proof strategy to that of [GS18]. In particular, our proof proceeds via a hybrid argument. In each hybrid we change the way the obfuscated circuit computes the garbled circuits for each step of the computation. Each garbled step circuit can be computed in three modes. The first mode is **real**, where the computations are just as in the real protocol. The second mode is **dummy**, where the output of the garbled circuit is constant and hardwired, but the same as in the real execution. The third mode is **sim**, which is similar to **real** mode, with the difference being that

the garbled circuit only outputs dummy values which are not the same as in the real execution. We cannot change directly from **real** mode to **sim** mode because at each step of the computation the labels from the previous step are visible to the adversary. Hence, we first need to change to **dummy** mode and then to **sim** mode. We show a set of rules that define a pebbling game, where the pebbles are represented by simulation slots. The aim of the game is to switch the pebbles from **real** (white pebbles) to **sim** (black pebbles), while minimizing the number of nodes in **dummy** (grey pebbles). Our objective is to minimize the number of grey pebbles at any point in time because the size of the obfuscated circuit grows with the number of simulation slots in **dummy** mode. Finally, with help of a pebbling strategy [GS18], we prove that our LFE construction is secure while having only a poly-logarithmic number of grey pebbles at any point in the simulation.

**Application: Witness Encryption for Turing Machines.** We show how our newly constructed LFE scheme allows us to construct witness encryption for Turing machines. To encrypt a message  $m$  with respect to a relation  $\mathcal{R}$ , the witness encryption algorithm computes the **crs** of the LFE and hashes  $d \leftarrow \text{LFE.Hash}(\text{crs}, \mathbf{M}_{\mathcal{R}})$ , where the Turing machine is defined as

$$M_{\mathcal{R}}(m, w) := \begin{cases} \text{return } m & \text{if } \mathcal{R}(x, w) = 1 \\ \text{return } \perp & \text{else} \end{cases} .$$

Then it returns the obfuscation of a circuit  $\text{obC} \leftarrow \text{iO}(\mathbf{C}_{x,m})$  where  $\mathbf{C}_{x,m}$  is defined as

$$\mathbf{C}_{x,m}(w) := \text{return LFE.Enc}(\text{crs}, d, (m, w)).$$

Given a witness  $w$ , one can recover  $m$  by querying the obfuscated circuit and evaluating the LFE decryption algorithm:

$$\begin{aligned} \text{LFE.Dec}(\text{crs}, \mathbf{M}_{\mathcal{R}}, \text{obC}(w)) &= \text{LFE.Dec}(\text{crs}, \mathbf{M}_{\mathcal{R}}, \mathbf{C}_{x,m}(w)) \\ &= \text{LFE.Dec}(\text{crs}, \mathbf{M}_{\mathcal{R}}, \text{LFE.Enc}(\text{crs}, d, (m, w))) \\ &= \mathbf{M}_{\mathcal{R}}(m, w) \\ &= m. \end{aligned}$$

Note that the size of the ciphertext is only dependent on the size of the witness  $w$ , the size of the message  $m$ , and the security parameter. Furthermore, the runtime of the decryption algorithm only depends on the runtime of the Turing machine computing  $\mathbf{M}_{\mathcal{R}}$ . Security follows via a standard puncturing argument.

**Application: ABE for Turing Machines.** We also sketch how to turn the above witness encryption into an ABE for Turing machines. This is a standard transformation [GGSW13] and therefore we only include an outline of the construction. To delegate a decryption key for a Turing machine  $\mathbf{M}$ , the authority computes a signature  $\sigma$  on the tuple  $(\text{crs}, \mathbf{d}_{\mathbf{M}})$ , where  $\mathbf{d}_{\mathbf{M}} \leftarrow \text{LFE.Hash}(\text{crs}, \mathbf{M})$  and



$\tilde{M}(x, m)$  returns  $m$  if and only if  $M(x) = 1$ . Then encrypting a message  $m$  with respect to an attribute  $x$  can be done by obfuscating

$C_{x,m}(\text{crs}, d, \sigma, x) : \mathbf{if} \text{ Verify}(\sigma, (\text{crs}, d)) = 1; \mathbf{return} \text{ LFE.Enc}(\text{crs}, d, (m, x)).$

Note that the runtime of the encryption algorithm (and consequently the size of the ciphertext) only depends on the size of the attribute  $x$  and the message  $m$ . Furthermore, the runtime of the decryption algorithm is only proportional to the runtime of the Turing machine  $M$ .

For additional details and further applications, we refer the reader to the full version.

### 1.3 Related Works

The notion of LFE was introduced in the work of Quach et al. [QWW18], in which they presented a construction for depth-bounded polynomial-size circuits from the learning with errors problem. Work by Pang, Chen, Fan, and Tang [PCFT20] extended the notion of (single-input) LFE to the multi-input settings, by additionally assuming the existence of indistinguishability obfuscation. Their protocol uses single-input LFE (and in particular the scheme from [QWW18]) generically. Thus, our scheme can be plugged into their work to obtain improved parameters.

Recent work by Agrawal and Roşie [AR21] shows a new construction of LFE with adaptive security (based on the ring learning with errors assumption). However, the scheme is limited to the computation of  $\text{NC}^1$  circuits. Another recent work by Naccache, Roşie, and Spignoli [NRS21] improves the concrete efficiency of LFE. In particular, the authors present a construction based on the LWE assumption with asymptotically smaller parameters than those used in [QWW18]. However, their construction is restricted to the class  $L/\text{poly}$ , i.e., the class of circuits that can be represented by branching programs of polynomial length.

## 2 Definitions

Let  $\lambda \in \mathbb{N}$  denote the security parameter. We say that a function  $\text{negl}(\cdot)$  is negligible if it vanishes faster than the inverse of any polynomial. Given a set  $S$ , we denote by  $s \leftarrow_{\$} S$  the uniform sampling from  $S$ . We say that an algorithm is PPT if it can be implemented by a probabilistic Turing machine running in time  $\text{poly}(\lambda)$ . Let  $X$  and  $Y$  denote two random variables and let  $\{X\}_{\lambda \in \mathbb{N}}$  and  $\{Y\}_{\lambda \in \mathbb{N}}$  be two distribution ensembles. We say that these distributions are computationally indistinguishable if for all PPT algorithms  $\mathcal{A}$ ,  $|\Pr_{x \leftarrow X_{\lambda}}[\mathcal{A}(x) = 1] - \Pr_{x \leftarrow Y_{\lambda}}[\mathcal{A}(x) = 1]| \leq \text{negl}(\lambda)$ . We denote this by  $X_{\lambda} \stackrel{c}{\approx} Y_{\lambda}$ . Let  $G_{par}$  denote a game, defined relative to a set of parameters  $par$ , where an adversary  $\mathcal{A}$  interacts with a challenger that answers oracle queries issued by  $\mathcal{A}$ . We denote the output of the game  $G_{par}$ , between a challenger and an adversary  $\mathcal{A}$ , as  $G_{par}^{\mathcal{A}}$ .  $\mathcal{A}$  is said to *win* the game if  $G_{par}^{\mathcal{A}} = 1$ . We define the advantage of  $\mathcal{A}$  in  $G_{par}$  as  $\text{Adv}_{par, \mathcal{A}}^G := \Pr[G_{par}^{\mathcal{A}} = 1]$ .

### 2.1 Laconic Function Evaluation for Turing Machines

Here, we adapt the definition of laconic function evaluation (LFE), a primitive recently introduced by Quach, Wichs, and Wee [QWW18], to that of LFE for Turing machines. The runtime of the Turing machine, denoted  $T$ , is publicly known and available to all parties. Without loss of generality we assume the Turing machine to be oblivious.

**Definition 1 (Laconic Function Evaluation for Turing Machines).** *A laconic function evaluation scheme  $\text{LFE} := (\text{LFE.Gen}, \text{LFE.Hash}, \text{LFE.Enc}, \text{LFE.Dec})$  for Turing machines is defined as the following tuple of PPT algorithms.*

- $\text{crs} \leftarrow \text{LFE.Gen}(1^\lambda, 1^N)$ : Given the security parameter  $1^\lambda$  and the block size  $1^N$  (encoded in unary), the generation algorithm returns a common reference string  $\text{crs}$ .
- $\text{d} \leftarrow \text{LFE.Hash}(\text{crs}, \text{M})$ : Given the common reference string  $\text{crs}$  and the description of a Turing machine  $\text{M}$ , the compression algorithm returns a digest  $\text{d}$ .
- $\text{c} \leftarrow \text{LFE.Enc}(\text{crs}, \text{d}, x)$ : Given the common reference string  $\text{crs}$ , a digest  $\text{d}$ , and a message  $x$ , the encoding algorithm returns a ciphertext  $\text{c}$ .
- $y \leftarrow \text{LFE.Dec}(\text{crs}, \text{M}, \text{c})$ : Given the common reference string  $\text{crs}$ , the description of a Turing machine  $\text{M}$ , and a ciphertext  $\text{c}$ , the decoding algorithm returns a message  $y$ .

For correctness, we require the encoding of an input with respect to the digest of a Turing machine, when decoded, to return the same result as evaluating the machine on the input. A more formal definition follows.

**Definition 2 (Correctness).** *A laconic function evaluation scheme  $\text{LFE} := (\text{LFE.Gen}, \text{LFE.Hash}, \text{LFE.Enc}, \text{LFE.Dec})$  for Turing machines is correct if for all  $\lambda \in \mathbb{N}$ ,  $N \in \mathbb{N}$ , for all Turing machines  $\text{M}$ , and all messages  $x$  it holds that*

$$\Pr \left[ \text{M}(x) = y \mid \begin{array}{l} \text{crs} \leftarrow \text{LFE.Gen}(1^\lambda, 1^N) \\ \text{d} \leftarrow \text{LFE.Hash}(\text{crs}, \text{M}) \\ \text{c} \leftarrow \text{LFE.Enc}(\text{crs}, \text{d}, x) \\ y \leftarrow \text{LFE.Dec}(\text{crs}, \text{M}, \text{c}) \end{array} \right] = 1,$$

where the probability is taken over the random coins of  $\text{LFE.Gen}$  and  $\text{LFE.Enc}$ .

The security notion captures the requirement that the encryption of a message  $x$  with respect to a compressed Turing machine  $\text{M}$  reveals nothing beyond  $\text{M}(x)$ .

**Definition 3 (Security: Sender-Privacy Against Semi-Honestbreak Receivers).** *A laconic function evaluation scheme  $\text{LFE} := (\text{LFE.Gen}, \text{LFE.Hash}, \text{LFE.Enc}, \text{LFE.Dec})$  for Turing machines is secure if there exists a PPT simulator*

$\text{Sim}_{\text{LFE}}$  such that for any stateful PPT adversaries  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  and  $N \in \mathbb{N}$  there exists a negligible function  $\text{negl}(\cdot)$  such that

$$\left| \Pr \left[ \mathcal{A}_2(c, \text{st}) = 1 \mid \begin{array}{l} \text{crs} \leftarrow \text{LFE.Gen}(1^\lambda, 1^N) \\ (x, M, \text{st}) \leftarrow \mathcal{A}_1(\text{crs}) \\ d \leftarrow \text{LFE.Hash}(\text{crs}, M) \\ c \leftarrow \text{LFE.Enc}(\text{crs}, d, x) \end{array} \right] - \Pr \left[ \mathcal{A}_2(c, \text{st}) = 1 \mid \begin{array}{l} \text{crs} \leftarrow \text{LFE.Gen}(1^\lambda, 1^N) \\ (x, M, \text{st}) \leftarrow \mathcal{A}_1(\text{crs}) \\ d \leftarrow \text{LFE.Hash}(\text{crs}, M) \\ c \leftarrow \text{Sim}_{\text{LFE}}(\text{crs}, d, M, M(x), T) \end{array} \right] \right| \leq \text{negl}(\lambda),$$

where the probability is taken over the random coins of  $\text{LFE.Gen}$ ,  $\mathcal{A}_1$ ,  $\text{LFE.Enc}$  and  $\text{Sim}_{\text{LFE}}$ . Here,  $T$  denotes the runtime of  $M(x)$  and  $\text{st}$  the state of  $\mathcal{A}$ .

An additional security property of an LFE scheme is that of *function hiding*, which captures the notion that the digest  $d \leftarrow \text{LFE.Hash}(\text{crs}, M)$  should hide the description of the Turing machine  $M$ . We note that our scheme can be generically transformed to satisfy function-hiding using the transformation of [QWW18]. The transformation uses 2-round 2PC based on OT and garbled circuits, and maintains the same asymptotic efficiency.

### 3 Laconic Function Evaluation for Turing Machines

In this section we will construct a laconic function evaluation scheme Fig. 4 with asymptotically optimal parameters.

**Notation.** We consider the case where the protocol computes a function  $F(m_A, m_B)$ , where  $m_A$  and  $m_B$  are the inputs of Alice and Bob, respectively. We assume that the function  $F(m_A, m_B)$  is computed by a Turing machine  $M$ , where  $m_A$  and  $m_B$  are given to  $M$  on two different input tapes. We assume without loss of generality that the Turing machine  $M$  is publicly known.<sup>2</sup> More formally,  $M$  denotes the 4-tape Turing machine consisting of two read-only input tapes, a read/write work tape, and a read/write output tape.  $M$  is described by the tuple  $(\Gamma, Q, \delta)$ , where  $\Gamma$  denotes the finite alphabet of  $M$  containing a blank symbol  $\square$  as well as a start symbol  $\triangleright$ , and the numbers 0 and 1;  $Q$  denotes a finite set of states containing a start state  $q_{\text{start}}$  and a halting state  $q_{\text{halt}}$ ; and  $\delta : Q \times \Gamma^4 \rightarrow Q \times \Gamma^2 \times \{\text{L}, \text{S}, \text{R}\}^4$  denotes the transition function. We assume that the transition function  $\delta$  of  $M$  is given by a circuit  $\mathbf{C}_M$ . It is going to be convenient for us to load the input  $m_B$  onto the working tape of the Turing machine. For the remainder of this description, we consider the working tape and the input tape of  $m_B$  as a single tape. Furthermore,  $M$  is an oblivious Turing machine, meaning its head movements do not depend on the input but only on the input length. Note, that by a classical result of Pippinger and Fischer,

<sup>2</sup> One can always make the function  $F$  private by including an encoding of  $F$  in the input of Alice and computing LFE of a universal Turing machine.

Turing machines can be simulated by an oblivious (and deterministic) Turing machine with only a logarithmic slowdown [PF79]. For convenience, we denote by  $\text{HeadPos}(i)$  the function that outputs the state  $\text{st}'$ , the write location on the working tape  $I_w$ , and the read locations  $I_r, J_r$  on the input tapes  $m_B$  and  $m_A$  respectively; all at step  $i$  of  $\mathbf{C}_M$ 's computation.

**Description.** Our scheme assumes the existence of:

- A symmetric encryption scheme  $\Pi := (\text{Sym.Gen}, \text{Sym.Enc}, \text{Sym.Dec})$  that is IND-CPA secure.
- An updatable laconic oblivious transfer  $\text{ULOT} := (\text{ULOT.Gen}, \text{ULOT.Hash}, \text{ULOT.Send}, \text{ULOT.Receive}, \text{ULOT.SendWriteRead}, \text{ULOT.ReceiveWriteRead})$  with sender privacy against semi-honest receivers.
- An indistinguishability obfuscator  $\text{iO}$ .
- A garbling scheme  $\text{GC} := (\text{GC.Garble}, \text{GC.Eval}, \text{GC.Input})$  with selective security.
- A puncturable pseudorandom function  $\text{PPRF} := (\text{PPRF.Gen}, \text{PPRF.Eval}, \text{PPRF.Punc})$ .

For convenience we make a few simplifying assumptions: (1) The Turing machine never writes to the same position twice (this does not affect its runtime, as we can just write to a new memory location every time) and (2) The input  $m_B$  is of length exactly  $N$ . Our scheme can be modified to handle the more general case but the description and the proof become somewhat more contrived.

The step circuit Fig. 2 handles the tasks performed at each step of  $\mathbf{M}$ 's computation. Namely, decrypting the secret input into  $\mathbf{C}_M$ , computing one step of  $\mathbf{C}_M$  and encrypting the output with a new key. Furthermore, after each step, additional outputs are used to specify a location in the database where the encrypted data is to be written using the updatable laconic oblivious transfer. The garbling step circuit Fig. 3 garbles each step circuit and generates the relevant labels and keys so that the garbled circuit can be evaluated.

We define the step circuit  $\mathbf{SC}_i$  as in Fig. 2. As inputs,  $\mathbf{C}_M$  takes the state  $\text{st} \in Q$  of the Turing machine  $\mathbf{M}$ , as well as two input blocks  $x_A \subseteq m_A$  and  $x_B = m_B$  both of size  $N$ . After evaluating the circuit on its inputs,  $\mathbf{C}_M$  returns a new state  $\text{st}' \in Q$ ; a write location  $I_w$  on the working tape, at which the next block of symbols  $y_B$  is written; a read location  $I_r$  on the input tape  $m_B$ ; a read location  $J_r$  on the input tape  $m_A$ ; and  $q = \perp$ , unless the halting state  $q_{\text{halt}}$  has been reached, in which case  $q$  is the only output of the computation.

Now we define the following circuit **GarbleSC**, which has the  $\text{crs}$  and a PRF seed  $s$  hardwired Fig. 3. It takes as input an index  $i$  and outputs a garbled circuit  $\mathbf{GC}^{(i)}$ .

We are now ready to present our laconic function evaluation protocol Fig. 4.

### 3.1 Correctness

The correctness of our LFE construction follows routinely from the correctness of its components, namely the indistinguishability obfuscator  $\text{iO}$ , the garbling

```

SCi [crs, ki, ki+1, lbsst, lbsA, lbsB] (st, zA, zB):
1. Parse (dA, xA) := zA
2. Parse (dB, x'B) := zB
3. Parse (lbsB[0], lbsB[1]) := lbsB
4. xB ← Sym.Dec(ki, x'B)
5. (st', Iw, yB, Ir, Jr, q) ← CM(st, xA, xB)
6. y'B ← Sym.Enc(ki+1, yB)
7. eA ← ULOT.Send(crs, dA, Jr, lbsA)
8. eB ← ULOT.SendWriteRead(crs, dB, Iw, y'B, lbsB[0], Ir, lbsB[1])
9.  $\widehat{\text{st}} \leftarrow \text{GC.Input}(\text{st}', \text{lbs}_{\text{st}})$ 
   return ( $\widehat{\text{st}}$ , Iw, y'B, Ir, Jr, eA, eB, q)

```

Fig. 2. Step Circuit.

```

GarbleSC[crs, s, k](i):
1. (lbsst || lbsA || lbsB || R) ← PPRF.Eval(s, i)
2. (lbs'st || lbs'A || lbs'B || ·) ← PPRF.Eval(s, i + 1)
3. (st, Iw, Ir, Jr) ← HeadPos(i)
4. (st', I'w, I'r, J'r) ← HeadPos(i + 1)
5. ki ← PPRF.Eval(k, Iw)
6. ki+1 ← PPRF.Eval(k, I'w)
7. C' ← SCi [crs, ki, ki+1, lbsst, lbs'A, lbs'B]
8. GC ← GC.Garble(1λ, C', (lbsst || lbsA || lbsB; R))
   return GC

```

Fig. 3. Garbling Step Circuit. The circuit is padded to the maximum size of  $\text{Sim}_{\text{GarbleSC}}$  [See proof of Theorem 3].

scheme GC, the updatable laconic oblivious transfer protocol ULOT, the symmetric encryption scheme  $\Pi$  and the puncturable pseudorandom function PPRF.

**Proposition 1 (Correctness).** *The Laconic Function Evaluation protocol in Fig. 4 is correct.*

*Proof of Proposition 1.* We prove the claim via an inductive argument. Let  $c_B^{(i)}$  denote the contents of the databases at the beginning of the  $i^{\text{th}}$  iteration of the while loop in LFE.Dec. Let  $\text{tr}'_i$  denote the transcript  $\text{tr}_i$  of  $M$ , except that we remove Alice's input tape  $m_A$ , and let  $T$  denote the runtime of  $M$ . We argue that  $\forall i \in \{1, \dots, T\}$ ,  $c_B^{(i)}$  block-wise decrypts to the transcript  $\text{tr}'_i$  at step  $i$  of  $M$ 's computation. We also show that at each  $i$ , the garbled input labels  $(\widehat{\text{st}}^{(i)} \parallel \widehat{z}_A^{(i)} \parallel \widehat{z}_B^{(i)})$  are a valid encoding of the state of the Turing machine  $M$ ,  $d_A$  the block of  $m_A$ , and  $d_B$  the block of  $c_B$  all in step circuit  $\text{SC}_i$ .

```

LFE.Gen ( $1^\lambda, 1^N$ ):
    1. Compute  $\text{crs} \leftarrow \text{ULOT.Gen}(1^\lambda, 1^N)$ 
       return  $\text{crs}$ 
LFE.Hash( $\text{crs}, m_A$ ):
    1. Compute  $(d_A, \widehat{m}_A) \leftarrow \text{ULOT.Hash}(\text{crs}, m_A)$ 
       return  $(d_A, \widehat{m}_A)$ 
LFE.Enc( $\text{crs}, d_A, m_B$ ):
    1. Choose two uniformly random PRF seeds  $(s, k)$ 
    2. Compute  $(\text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B \parallel R) \leftarrow \text{PPRF.Eval}(s, 1)$ 
    3. Compute  $k_1 \leftarrow \text{PPRF.Eval}(k, 1)$ 
    4. Compute  $\text{obG} \leftarrow \text{iO}(\text{GarbleSC}[\text{crs}, s, k])$ 
    5. Block-wise encrypt  $c_B \leftarrow \text{Sym.Enc}(k_1, m_B)$ 
    6. Compute  $(d_B, \widehat{c}_B) \leftarrow \text{ULOT.Hash}(\text{crs}, c_B)$ 
    7. Set  $\text{st} \leftarrow 0^N$ 
    8. Set  $z_A \leftarrow (d_A, 0^N)$ 
    9. Set  $z_B \leftarrow (d_B, c_B)$ 
    10. Compute  $\widehat{\text{st}} \leftarrow \text{GC.Input}(\text{st}, \text{lbs}_{\text{st}})$ 
    11. Compute  $\widehat{z}_A \leftarrow \text{GC.Input}(z_A, \text{lbs}_A)$ 
    12. Compute  $\widehat{z}_B \leftarrow \text{GC.Input}(z_B, \text{lbs}_B)$ 
    13. Set  $c \leftarrow (\widehat{c}_B, \text{obG}, \widehat{\text{st}}, \widehat{z}_A, \widehat{z}_B)$ 
       return  $c$ 
LFE.Dec( $\text{crs}, m_A, c$ ):
    1. Parse  $(\widehat{c}_B, \text{obG}, \widehat{\text{st}}, \widehat{z}_A, \widehat{z}_B) := c$ 
    2. Set  $m_B^{(1)} \leftarrow \widehat{c}_B, \widehat{\text{st}}^{(1)} \leftarrow \widehat{\text{st}}, \widehat{z}_A^{(1)} \leftarrow \widehat{z}_A, \widehat{z}_B^{(1)} \leftarrow \widehat{z}_B$ 
    3. Set  $i := 1$ 
    4.  $q := \perp$ 
    5. while true do
       if  $q \neq \perp$  then
           return  $q$ 
       Compute  $\text{GC}^{(i)} \leftarrow \text{obG}^{(i)}$ 
       Compute  $(\widehat{\text{st}}^{(i+1)} \parallel I_w \parallel m_B^{(i+1)} \parallel I_r \parallel J_r \parallel e_A \parallel e_B \parallel q) \leftarrow$ 
        $\text{GC.Eval}(\text{GC}^{(i)}, (\widehat{\text{st}}^{(i)} \parallel \widehat{z}_A^{(i)} \parallel \widehat{z}_B^{(i)}))$ 
       Compute  $\widehat{z}_A^{(i+1)} \leftarrow \text{ULOT.Receive}^{m_A}(\text{crs}, e_A, J_r)$ 
       Compute  $\widehat{z}_B^{(i+1)} \leftarrow \text{ULOT.ReceiveWriteRead}^{\widehat{c}_B}(\text{crs}, I_w, m_B^{(i)}, e_B, I_r)$ 
       Set  $i := i + 1$ 
    
```

**Fig. 4.** Laconic Function Evaluation Protocol.

The base case, when  $i = 1$ , follows trivially. Initially, the database  $c_B^{(1)}$  contains a block-wise encryption of  $m_B$  [step 5 of LFE.Enc]. In step 9 of LFE.Enc  $x'_B$  is set to  $c_B^{(1)}$ , i.e.  $x'_B$  contains  $m_B$  and the content of the (empty) worktape. Similarly,  $x_A$  is also initialised to  $0^N$  in step 8 of LFE.Enc. Hence, the tran-

script  $\text{tr}'_1$  consists of the input tape  $m_B$  concatenated with an empty working tape and the state. Thus,  $\text{Sym.Dec}(k_1, c_B^{(1)}) = \text{tr}'_1$ . The garbled input labels  $(\widehat{\text{st}}^{(1)} \parallel \widehat{z}_A^{(1)} \parallel \widehat{z}_B^{(1)})$  are passed to  $\text{LFE.Dec}$  in the ciphertext.

By the inductive hypothesis we assume that the database  $c_B^{(i-1)}$  block-wise decrypts to give  $\text{tr}'_{i-1}$ . We now show that  $\text{Sym.Dec}(k_i, c_B^{(i)}) = \text{tr}'_i$ . In the  $i^{\text{th}}$  iteration of the while loop in  $\text{LFE.Dec}$ ,  $\mathbf{SC}_i$  is evaluated by  $\mathbf{GC}^{(i)}$ . Due to the correctness of the indistinguishability obfuscator  $\text{iO}$ , the obfuscated garbling step circuit  $\text{obG}$  can be correctly evaluated on input  $i$ , and  $\mathbf{GC}^{(i)}$  is given by

$$\begin{aligned} \mathbf{GC}^{(i)} &= \text{obG}(i) \\ &= \text{iO}(\mathbf{GarbleSC}[\text{crs}, s, k](i)) \\ &= \text{GC.Garble}\left(1^\lambda, \mathbf{SC}_i[\text{crs}, k_i, k_{i+1}, \text{lbs}'_{\text{st}}, \text{lbs}'_A, \text{lbs}'_B](\cdot, \cdot, \cdot), \text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B; R\right). \end{aligned}$$

By the induction hypothesis, the garbled input labels  $(\widehat{\text{st}}^{(i)} \parallel \widehat{z}_A^{(i)} \parallel \widehat{z}_B^{(i)})$  are a valid encoding of the state of the Turing machine  $\mathbf{M}$ ,  $d_A$  and the block of  $m_A$ , and  $d_B$  and the block of  $c_B$  all in step circuit  $\mathbf{SC}_i$ . In  $\mathbf{SC}_i$ , the decryption of  $x_B^{(i)}$  gives  $x_B^{(i)}$ . After running  $\mathbf{C}_M$ ,  $y_B^{(i)}$  is then written to the work tape at  $I_w^{(i)}$ , and encrypted to  $y_B^{\prime(i)}$ . By the correctness of updatable laconic oblivious transfer,  $\text{ULOT.SendWriteRead}$  specifies  $y_B^{\prime(i)}$  to be written to a database and in step 5 of  $\text{LFE.Dec}$ ,  $y_B^{\prime(i)}$  is written to  $c_B$  at position  $I_w^{(i)}$ . Therefore,  $\text{Sym.Dec}(k_i, c_B^{(i)}) = \text{tr}'_{i-1}$  with  $y_B^{(i)}$  written on the work tape at  $I_w^{(i)}$ . I.e.,  $\text{Sym.Dec}(k_i, c_B^{(i)}) = \text{tr}'_i$ . Furthermore, the garbled input labels  $\widehat{z}_A^{(i+1)}$  and  $\widehat{z}_B^{(i+1)}$  are given by

$$\begin{aligned} \widehat{z}_A^{(i+1)} &= \text{ULOT.Receive}^{m_A^{(i)}}\left(\text{crs}, e_A^{(i)}, J_r^{(i)}\right) \\ &= \text{ULOT.Receive}^{m_A^{(i)}}\left(\text{crs}, \text{ULOT.Send}\left(\text{crs}, d_A, J_r^{(i)}, \text{lbs}_A\right), J_r^{(i)}\right), \end{aligned}$$

and

$$\begin{aligned} \widehat{z}_B^{(i+1)} &= \text{UL OT.ReceiveWriteRead}^{\widehat{c}_B^{(i)}}\left(\text{crs}, I_w^{(i)}, m_B^{(i)}, e_B^{(i)}, I_r^{(i)}\right) \\ &= \text{UL OT.ReceiveWriteRead}^{\widehat{c}_B^{(i)}}\left(\text{crs}, I_w^{(i)}, m_B^{(i)}, \right. \\ &\quad \left. \text{ULOT.SendWriteRead}\left(\text{crs}, d_B, I_w^{(i)}, y_B^{\prime(i)}, \text{lbs}_{B[0]}, I_r, \text{lbs}_{B[1]}\right), I_r^{(i)}\right), \end{aligned}$$

respectively.

### 3.2 Proof of Security

We will now establish sender simulation security for our protocol, and start by stating the main security theorem.

**Theorem 3 (Security).** *Assume that  $iO$  is an indistinguishability obfuscator,  $(GC.Garble, GC.Input, GC.Eval)$  is simulation secure,  $(ULOT.Gen, ULOT.Hash, ULOT.Send, ULOT.Receive, ULOT.SendWriteRead, ULOT.ReceiveWriteRead)$  has sender privacy against semi-honest receivers,  $(Sym.Gen, Sym.Enc, Sym.Dec)$  is IND-CPA secure, and that  $(PPRF.Gen, PPRF.Eval, PPRF.Punc)$  is a puncturable pseudorandom function. Then  $(LFE.Gen, LFE.Hash, LFE.Enc, LFE.Dec)$  has sender privacy against semi-honest receivers.*

To prove the security of our construction we use a similar proof strategy to that of [GS18]. In particular, our proof will proceed via a hybrid argument. In each hybrid we change the way the circuit  $obG$  computes the garbled circuits  $GC^{(i)}$ . Each garbled step circuit  $GC^{(i)}$  can be computed in three modes Fig. 8. The first mode is *real*, where the computations are just as in the real protocol. The second mode is *dummy*, where the output of the garbled circuit is constant and hardwired, but the same as in the real execution Fig. 5 and 6. The third mode is *sim*, which is similar to *real* mode, with the difference being that the garbled circuit only outputs dummy values which are not the same as in the real execution Fig. 2.

Both garbled circuits in *real* and *dummy* mode will keep the intermediate states and memory consistent (recall that the memory is accessed via an updatable laconic OT). On the other hand, a garbled circuit in *sim* mode will only output the dummy state and perform dummy read and writes to memory. Garbled circuits in *real* and *sim* mode are computed on-the-fly by  $obG$ , whereas circuits in *dummy* need to be hardwired into  $obG$ . As a result, the size of  $obG$  depends on the maximum number of *dummy* circuits needed in any given hybrid.

We will briefly discuss the necessary conditions under which we can switch the mode of a garbled step circuit. The first garbled circuit in the in line  $GC^{(1)}$  can always be switched from *real* to *dummy* or vice versa, provided there is a free simulation slot available, i.e., the number of currently simulated garbled circuits is less than some maximum amount  $t$ . For any other garbled circuit  $GC^{(i)}$ , we can switch its mode from *real* to *dummy* or vice versa, given that the circuit  $GC^{(i-1)}$  is in *dummy* mode and a simulation slot is available. To switch a node into *sim* mode, we require that its successor node is in *sim* mode and that its predecessor is in *dummy* mode. In the case of the first node we only have the requirement for its successor node and for the last node we only have the requirement for its predecessor.

These rules define a pebbling game, where we identify pebbles as simulation slots. The goal of the game is to switch the nodes from *real* (white pebbles) to *sim* (black pebbles), while minimizing the number of nodes in *dummy* (grey pebbles). To win the game, we can use the same pebbling strategy as in [GS18], where  $\mathcal{O}(\log(T))$  pebbles suffice to set a pebble at the last node (with index  $T$ ) in  $\text{poly}(T)$  steps. Consequently, with this strategy we only need to simulate  $\mathcal{O}(\log(T)) = \mathcal{O}(\lambda)$  nodes in any given hybrid. We refer the reader to the works of [GPSZ17] and [GS18] for an optimal strategy for the pebbling game. For the sake of completeness we state the main Lemmas here.



**Lemma 1** ([GPSZ17]). *For any  $p \in \mathbb{Z}$ , such that  $n + 1 \leq p \leq n + 2^k - 1$ , it is possible to make  $\mathcal{O}((p - n)^{\log_2 3}) \approx \mathcal{O}((p - n)^{1.585})$  moves and get a black pebble at position  $p$  using  $k$  gray pebbles.*

**Lemma 2** ([GS18]). *For any  $T \in \mathbb{N}$ , there exists a strategy for pebbling the line graph  $\{1, \dots, T\}$  according to rules  $\mathfrak{A}$  and  $\mathfrak{B}$  by using at most  $\log(T)$  grey pebbles and making  $\text{poly}(\lambda)$  moves.*

Thus, our proof strategy will proceed as follows. First we will use the above pebbling argument to switch the last node, i.e. the node with index  $T$  to *sim* mode. This will take  $\text{poly}(T)$  steps. Next, we will again use the same pebbling argument to switch node  $T - 1$  to *sim* mode. This will take  $\text{poly}(T - 1) = \text{poly}(T)$  steps. Consequently, we replace nodes  $T - 2, T - 3, \dots, 2, 1$  with *sim* nodes, in this order. In total, this will require  $T \cdot \text{poly}(T) = \text{poly}(T)$  steps. In the very last hybrid, we will replace the encryption of that database  $m_B$  by an encryption of 0. Once all pebbles (step circuits) are in *sim* mode, and the encryption of  $m_B$  has been replaced with the encryption of 0, this corresponds to the simulator  $\text{Sim}_{\text{LFE}}$ , which takes as input the *crs*, *d*, the machine  $\mathbb{M}$ , the output  $\mathbb{M}(x)$  and the time bound  $T$ . The simulator then outputs the ciphertext  $c$ . As a result, the view of the adversary, in this last hybrid, is independent of the sender input  $m_B$ . Hence, we can use this hybrid to simulate the view of a semi-honest receiver by only using the receiver’s output. The full proof of Theorem 3 follows from that of two lemmas [Lemma 3, Lemma 4].

**Circuit Configuration.** A circuit configuration *conf* consists of a subset of garbling step circuits in *dummy* mode as well as an index  $i^* \in \{1, \dots, T\}$  denoting the garbling step circuit to be changed by the rule.

**Rules of Indistinguishability.** We define the rules of indistinguishability (which determine the configurations in the pebbling game) below.

**Rule  $\mathfrak{A}$ :** Rule  $\mathfrak{A}$  dictates when a garbling step circuit can be indistinguishably changed from *real* mode to *dummy* mode. Let *conf* and *conf'* be two valid configurations and  $i^*$  be an index of the garbling step circuit, such that:

- Index  $i^*$  is changed from *real* mode to *dummy* mode, and there are no indices in *sim* mode to the left of  $i^*$ .
- Index  $i^*$  is either the first or its predecessor is in *dummy* mode.
- The garbling step circuits in *sim* mode remain unchanged.

In Lemma 3 we show that for two valid circuit configurations *conf* and *conf'*, satisfying the above constraints, the two distributions  $\mathcal{H}_{\text{conf}}$  and  $\mathcal{H}_{\text{conf}'}$  are computationally indistinguishable.

**Rule  $\mathfrak{B}$ :** Rule  $\mathfrak{B}$  dictates when a step circuit can be indistinguishably changed from *dummy* mode to *sim* mode. Let *conf* and *conf'* be two valid configurations and  $i^*$  be an index of the garbling step circuit, such that:

- Index  $i^*$  is changed from *dummy* mode to *sim* mode.
- Index  $i^*$  is either the last or its predecessor is in *dummy* mode.
- The garbling step circuits in *real* mode remain unchanged.

```

SCi*dummy[crs, ki*, ki*+1, lbsst, lbsA, lbsB](st, zA, zB):
1. Parse (dA, xA) := zA
2. Parse (dB, x'B) := zB
3. Parse (lbsB[0], lbsB[1]) := lbsB
4. xB ← Sym.Dec(ki*, xB)
5. (st', Iw, yB, Ir, Jr, q) ← CM(st, xA, xB)
6. y'B ← Sym.Enc(ki*+1, yB)
7. d*B ← ULOT.Hash(crs, m*B)
8. eA ← SimULOT.S(crs, mA, Jr, GC.Input(lbsA, mA[Jr]))
9. eB ← SimULOT.WR(crs, mB, Iw, y'B, GC.Input(lbsB[0], d*B), Ir,
GC.Input(lbsB[1], m*B[Ir]))
10. st̂ ← GC.Input(st', lbsst)
    return (st̂, Iw, y'B, Ir, Jr, eA, eB, q)

```

**Fig. 5.** Step Circuit in dummy mode. Let  $m_B^*$  denote the database that is identical to  $m_B$  except that  $m_B^*[I_w] = y'_B$ .

In Lemma 4 we show that for two valid circuit configurations  $\text{conf}$  and  $\text{conf}'$ , satisfying the above constraints, the two distributions  $\mathcal{H}_{\text{conf}}$  and  $\mathcal{H}_{\text{conf}'}$  are computationally indistinguishable.

```

GCi*dummy[crs, s, k]:
1. (lbsst || lbsA || lbsB || R) ← PPRF.Eval(s, i)
2. (lbs'st || lbs'A || lbs'B || ·) ← PPRF.Eval(s, i + 1)
3. (st, Iw, Ir, Jr) ← HeadPos(i)
4. (st', I'w, I'r, J'r) ← HeadPos(i + 1)
5. ki ← PPRF.Eval(k, Iw)
6. ki+1 ← PPRF.Eval(k, I'w)
7. Lst ← GC.Input(lbsst, st(i*))
8. LA ← GC.Input(lbsA, zA(i*))
9. LB ← GC.Input(lbsB, zB(i*))
10. out ← SCi*dummy[crs, ki*, ki*+1, lbs'st, lbs'A, lbs'B](st(i*), zA(i*), zB(i*))
11. GC ← SimGC(1λ, 1|SCi*dummy|, out, (Lst || LA || LB; R))
    return GC

```

**Fig. 6.** Garbling Step Circuit in dummy mode.

```

SCi*sim[crs, ki*, ki*+1, lbsst, lbsA, lbsB](st, zA, zB):
1. Parse (dA, xA) := zA
2. Parse (dB, x'B) := zB
3. Parse (lbsB[0], lbsB[1]) := lbsB
4. (st', Iw, Ir, Jr) ← HeadPos(i*)
5. y'B ← Sym.Enc(ki*+1, 0)
6. if i* = T then
   q := C(x)
7. else
   q := ⊥
8. eA ← ULOT.Send(crs, dA, Jr, lbsA)
9. eB ← ULOT.SendWriteRead(crs, dB, Iw, y'B, lbsB[0], Ir, lbsB[1])
10. st ← GC.Input(st', lbsst)
    return (st, Iw, y'B, Ir, Jr, eA, eB, q)

```

Fig. 7. Step Circuit in sim mode.

```

SimGarbleSC[crs, s](i*):
1. if i* ∈ dummy then
   return GCi*dummy[crs, s, k]
2. else
   (lbsst || lbsA || lbsB || R) ← PPRF.Eval(s, i)
   (lbsst || lbs'A || lbs'B || ·) ← PPRF.Eval(s, i + 1)
   (st', Iw, Ir, Jr) ← HeadPos(i)
   (st', I'w, I'r, J'r) ← HeadPos(i + 1)
   ki ← PPRF.Eval(k, Iw)
   ki+1 ← PPRF.Eval(k, I'w)
3. if i* ∈ real then
   Set C' ← SCi*[crs, ki*, ki*+1, lbs'st, lbs'A, lbs'B]
4. if i* ∈ sim then
   Set C' ← SCi*sim[crs, ki*, ki*+1, lbs'st, lbs'A, lbs'B]
   GC ← GC.Garble(1λ, C', (lbsst || lbsA || lbsB; R))
return GC

```

Fig. 8. Garbling Step Circuit in real, dummy and sim mode.

### 3.3 Proof of Indistinguishability for the Rules

#### Implementing Rule $\mathfrak{A}$

**Lemma 3 (Rule  $\mathfrak{A}$ ).** *Let conf and conf' be two valid circuit configurations satisfying the constraints of rule  $\mathfrak{A}$ . Assume that iO is an indistinguishability obfuscator, GC is simulation secure, ULOT has sender privacy against semi-honest receivers, and that PPRF is a puncturable pseudorandom function. Then,*

for the two distribution ensembles  $\{\mathcal{H}_{\text{conf},\lambda}\}_{\lambda \in \mathbb{N}}$  and  $\{\mathcal{H}_{\text{conf}',\lambda}\}_{\lambda \in \mathbb{N}}$  it holds that

$$\left| \Pr_{c \leftarrow \mathcal{H}_{\text{conf},\lambda}} [\mathcal{A}(1^\lambda, c) = 1] - \Pr_{c \leftarrow \mathcal{H}_{\text{conf}',\lambda}} [\mathcal{A}(1^\lambda, c) = 1] \right| \leq \text{negl}(\lambda).$$

*Proof of Lemma 3.* We prove this with help of a hybrid argument.

$\mathcal{H}_{\text{conf},\lambda}$ : The garbling step circuit is in real mode.

$\mathcal{H}_1$ : Instead of hardwiring the PPRF key  $s$  into  $\text{Sim}_{\text{GarbleSC}}$ , we hardwire the key  $s\{i^*\} \leftarrow \text{PPRF.Punc}(s, i^*)$ , that is punctured at  $i^*$ . Since we cannot evaluate  $\text{PPRF.Eval}(s\{i^*\}, i^*)$ , we additionally hardwire the labels and key that are output by  $\text{PPRF.Eval}(s, i^*)$  into  $\text{Sim}_{\text{GarbleSC}}$ .

$$(\text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B \parallel R) \leftarrow \text{PPRF.Eval}(s, i^*)$$

To be able to use the security of  $\text{iO}$ , the size of  $\text{GarbleSC}$  is padded to be the same size as  $\text{Sim}_{\text{GarbleSC}}$ .

*Claim* ( $\mathcal{H}_{\text{conf}} \rightarrow \mathcal{H}_1$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_{\text{conf}}$  and  $\mathcal{H}_1$  is:

$$\text{Adv}_{\mathcal{A}_1}^{\mathcal{H}_{\text{conf}} \rightarrow \mathcal{H}_1} \leq \text{Adv}_{\text{iO}, \mathcal{B}_1}^{\text{iO-sec}}.$$

$\mathcal{H}_{\text{conf}} \rightarrow \mathcal{H}_1$ . The proof relies on the security of the indistinguishability obfuscator  $\text{iO}$  to be able to switch the PPRF key and hardwire the labels. The reduction  $\mathcal{B}_1$  gets a bit  $b$  from the adversary  $\mathcal{A}_1$ , where  $b = 0$  if the obfuscated circuit is as described in  $\mathcal{H}_{\text{conf}}$  and  $b = 1$  if the obfuscated circuit is as described in  $\mathcal{H}_1$ . If  $\mathcal{A}_1$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_1$  wins the  $\text{iO-sec}$  game with greater than  $\epsilon$  probability.  $\square$

$\mathcal{H}_2$ : As opposed to using the labels output by  $\text{PPRF.Eval}(s, i^*)$ , we sample a string  $u$  from the uniform distribution  $U_\lambda$ .

*Claim* ( $\mathcal{H}_1 \rightarrow \mathcal{H}_2$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_1$  and  $\mathcal{H}_2$  is:

$$\text{Adv}_{\mathcal{A}_2}^{\mathcal{H}_1 \rightarrow \mathcal{H}_2} \leq \text{Adv}_{\text{PPRF}, \mathcal{B}_2}^{\text{PPRF-rand}}.$$

$\mathcal{H}_1 \rightarrow \mathcal{H}_2$ . The proof relies on the pseudorandomness property of PPRF, to be able to switch the output of  $\text{PPRF.Eval}(s, i^*)$  with  $u$ . The reduction  $\mathcal{B}_2$  gets a bit  $b$  from the adversary  $\mathcal{A}_2$ , where  $b = 0$  if the output of  $\text{PPRF.Eval}(s, i^*)$  is used, and  $b = 1$  if the uniform string is used. If  $\mathcal{A}_2$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_2$  wins the PPRF-rand game with greater than  $\epsilon$  probability.  $\square$

$\mathcal{H}_3$ : Since each label is computed twice, once in step  $i^* - 1$  and once in step  $i^*$ , we now remove the following labels at step  $i^* - 1$ ;

$$\begin{aligned} & \text{lbs}_{\text{st}} \setminus \text{GC.Input} \left( \text{lbs}_{\text{st}}, \text{st}^{(i^*-1)} \right) \\ & \text{lbs}_A \setminus \text{GC.Input} \left( \text{lbs}_A, z_A^{(i^*-1)} \right) \\ & \text{lbs}_B \setminus \text{GC.Input} \left( \text{lbs}_B, z_B^{(i^*-1)} \right). \end{aligned}$$

I.e., those used in steps 8–10 in  $\mathbf{SC}_{i^*-1}^{\text{dummy}}$ . This is possible, since by the constraints of rule  $\mathfrak{A}$ , the previous step is known to be in dummy mode.

*Claim* ( $\mathcal{H}_2 \rightarrow \mathcal{H}_3$ ). The distributions  $\mathcal{H}_2$  and  $\mathcal{H}_3$  are identical.

$\mathcal{H}_2 \rightarrow \mathcal{H}_3$ . We note that  $\mathbf{SC}_{i^*}^{\text{dummy}}$  is not executed in the obfuscated circuit, but rather computed locally by the simulator. The output is hardwired in the obfuscated circuit, and we are simply removing unused variables.  $\square$

$\mathcal{H}_4$ : We hardwire the output out of

$$\text{GC.Garble} \left( 1^\lambda, \mathbf{SC}_{i^*} \left[ \text{crs}, k_{i^*}, k_{i^*+1}, \text{lbs}'_{\text{st}}, \text{lbs}'_A, \text{lbs}'_B \right], (\text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B; R) \right)$$

into  $\text{Sim}_{\text{GarbleSC.iO}}$  reduction.

*Claim* ( $\mathcal{H}_3 \rightarrow \mathcal{H}_4$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_3$  and  $\mathcal{H}_4$  is:

$$\text{Adv}_{\mathcal{A}_4}^{\mathcal{H}_3 \rightarrow \mathcal{H}_4} \leq \text{Adv}_{\text{iO}, \mathcal{B}_4}^{\text{iO-sec}}.$$

$\mathcal{H}_3 \rightarrow \mathcal{H}_4$ . The proof relies on the security of the indistinguishability obfuscator iO to be able to hardwire the output of the garbling scheme. The reduction  $\mathcal{B}_4$  gets a bit  $b$  from the adversary  $\mathcal{A}_4$ , where  $b = 0$  if the obfuscated circuit is as described in  $\mathcal{H}_3$  and  $b = 1$  if the obfuscated circuit is as described in  $\mathcal{H}_4$ . If  $\mathcal{A}_4$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_4$  wins the iO-sec game with greater than  $\epsilon$  probability.  $\square$

$\mathcal{H}_5$ : We simulate the garbling step circuit, as

$$\text{GC} \leftarrow \text{Sim}_{\text{GC}} \left( 1^\lambda, 1^{|\mathbf{SC}_{i^*}|}, \text{out}, (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right),$$

where  $\text{out} \leftarrow \mathbf{SC}_{i^*} \left[ \text{crs}, k_{i^*}, k_{i^*+1}, \text{lbs}'_{\text{st}}, \text{lbs}'_A, \text{lbs}'_B \right] \left( \text{st}^{(i^*+1)}, z_A^{(i^*+1)}, z_B^{(i^*+1)} \right)$ , and  $(\text{lbs}'_{\text{st}} \parallel \text{lbs}'_A \parallel \text{lbs}'_B \parallel R') \leftarrow \text{PPRF.Eval}(\text{s}\{i^*\}, i^* + 1)$ . Recall that  $\text{st}^{(i^*+1)}$ ,  $z_A^{(i^*+1)}$ , and  $z_B^{(i^*+1)}$  denote the state of the Turing machine  $\mathbf{M}$ ; the digest  $\text{d}_A$  and the input block  $x_A$ ; as well as the digest  $\text{d}_B$  and the encrypted input block  $x_B$ , respectively, each at step  $i^* + 1$  of the computation.

*Claim* ( $\mathcal{H}_4 \rightarrow \mathcal{H}_5$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_4$  and  $\mathcal{H}_5$  is:

$$\text{Adv}_{\mathcal{A}_5}^{\mathcal{H}_4 \rightarrow \mathcal{H}_5} \leq \text{Adv}_{\text{GC}, \mathcal{B}_5}^{\text{GC-sec}}.$$

$\mathcal{H}_4 \rightarrow \mathcal{H}_5$ . The proof relies on the selective simulation security of the garbling scheme GC to be able to simulate the garbling step circuit. The reduction  $\mathcal{B}_5$  gets a bit  $b$  from the adversary  $\mathcal{A}_5$ , where  $b = 0$  if  $\mathcal{A}_5$  identified

$$\left\{ \text{GC.Garble} \left( 1^\lambda, \mathbf{SC}_{i^*} \left[ \text{crs}, k_{i^*}, k_{i^*+1}, \text{lbs}'_{\text{st}}, \text{lbs}'_A, \text{lbs}'_B \right], (\text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B; R) \right), (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right\}$$

and  $b = 1$  if  $\mathcal{A}_5$  identified

$$\left\{ \text{Sim}_{\text{GC}} \left( 1^\lambda, 1^{|\text{SC}_{i^*}|}, \text{out}, (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right), (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right\}.$$

If  $\mathcal{A}_5$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_5$  wins the GC-sec game with greater than  $\epsilon$  probability. □

$\mathcal{H}_6$ : We simulate the ULOT.Send ciphertext as  $e_A \leftarrow \text{Sim}_{\text{ULOT.S}}(\text{crs}, m_A, J_r, \text{GC.Input}(\text{lbs}_A, m_{A[J_r]}))$ . Recall that  $m_{A[J_r]}$  denotes M's input tape  $m_A$  at read location  $J_r$ , all at step  $i^*$ .

*Claim* ( $\mathcal{H}_5 \rightarrow \mathcal{H}_6$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_5$  and  $\mathcal{H}_6$  is:

$$\text{Adv}_{\mathcal{A}_6}^{\mathcal{H}_5 \rightarrow \mathcal{H}_6} \leq \text{Adv}_{\text{ULOT}, \mathcal{B}_6}^{\text{SenPriExpt}}.$$

$\mathcal{H}_5 \rightarrow \mathcal{H}_6$ . The proof relies on the semi-honest sender privacy of ULOT to be able to simulate the ciphertext. The reduction  $\mathcal{B}_6$  gets a bit  $b$  from the adversary  $\mathcal{A}_6$ , where  $b = 0$  if  $\mathcal{A}_6$  identified the ciphertext as

$$\{ \text{ULOT.Send}(\text{crs}, d_A, J_r, \text{lbs}_A) \}$$

and  $b = 1$  if  $\mathcal{A}_6$  identified the ciphertext as

$$\left\{ \text{Sim}_{\text{ULOT.S}}(\text{crs}, m_A, J_r, \text{GC.Input}(\text{lbs}_A, m_{A[J_r]})) \right\}.$$

If  $\mathcal{A}_6$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_6$  wins the SenPriExpt game with greater than  $\epsilon$  probability. □

$\mathcal{H}_7$ : We simulate the ULOT.SendWriteRead ciphertext as  $e_B \leftarrow \text{Sim}_{\text{ULOT.WR}}(\text{crs}, m_B, I_w, y'_B, \text{GC.Input}(\text{lbs}_{B[0]}, d_B^*), I_r, \text{GC.Input}(\text{lbs}_{B[1]}, m_{B^*[I_r]}^*))$ . Here,  $m_B^*$  denotes the database that is identical to  $m_B$  except that  $m_B^*[I_w] = y'_B$ , and  $d_B^* \leftarrow \text{ULOT.Hash}(\text{crs}, m_B^*)$ . Recall that  $I_w$ ,  $I_r$ , and  $y'_B$  denote the write location on the working tape; the read location on the input tape  $m_B$ ; and the encrypted block of symbols that are output by M, respectively, all step  $i^*$  of the computation.

*Claim* ( $\mathcal{H}_6 \rightarrow \mathcal{H}_7$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_6$  and  $\mathcal{H}_7$  is:

$$\text{Adv}_{\mathcal{A}_7}^{\mathcal{H}_6 \rightarrow \mathcal{H}_7} \leq \text{Adv}_{\text{ULOT}, \mathcal{B}_7}^{\text{WriReaSenPriExpt}}.$$

$\mathcal{H}_6 \rightarrow \mathcal{H}_7$ . The proof relies on the semi-honest sender privacy for writes and reads of ULOT to be able to simulate the ciphertext. The reduction  $\mathcal{B}_7$  gets a bit  $b$  from the adversary  $\mathcal{A}_7$ , where  $b = 0$  if  $\mathcal{A}_7$  identified the ciphertext as

$$\{ \text{ULOT.SendWriteRead}(\text{crs}, d_B, I_w, y'_B, \text{lbs}_{B[0]}, I_r, \text{lbs}_{B[1]}) \}$$

and  $b = 1$  if  $\mathcal{A}_7$  identified the ciphertext as

$$\left\{ \text{Sim}_{\text{ULOT.WR}} \left( \text{crs}, m_B, I_w, y'_B, \text{GC.Input}(\text{lbs}_{B[0]}, d_B^*), I_r, \right. \right. \\ \left. \left. \text{GC.Input}(\text{lbs}_{B[1]}, m_{B[I_r]}^*) \right) \right\}.$$

If  $\mathcal{A}_7$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_7$  wins the  $\text{WriReaSenPriExpt}$  game with greater than  $\epsilon$  probability.

$\mathcal{H}_8 - \mathcal{H}_{10}$ : Finally, we revert the changes made in  $\mathcal{H}_1 - \mathcal{H}_3$ . Here, the indistinguishability between  $\mathcal{H}_8 - \mathcal{H}_{10}$  follows analogous to that of  $\mathcal{H}_1 - \mathcal{H}_3$ .

$\mathcal{H}_{\text{conf}'_\lambda}$ : The step circuit is in dummy mode. ■

This concludes the proof of Lemma 3.

### Implementing Rule $\mathfrak{B}$

**Lemma 4 Rule  $\mathfrak{B}$** . *Let  $\text{conf}$  and  $\text{conf}'$  be two valid circuit configurations satisfying the constraints of rule  $\mathfrak{B}$ . Assume that  $\text{iO}$  is an indistinguishability obfuscator,  $\text{GC}$  is simulation secure,  $\text{ULOT}$  has sender privacy against semi-honest receivers, and that  $\text{PPRF}$  is a puncturable pseudorandom function. Then, for the two distribution ensembles  $\{\mathcal{H}_{\text{conf}'_\lambda}\}_{\lambda \in \mathbb{N}}$  and  $\{\mathcal{H}_{\text{conf}_\lambda}\}_{\lambda \in \mathbb{N}}$  it holds that*

$$\left| \Pr_{c \leftarrow \mathcal{H}_{\text{conf}'_\lambda}} [\mathcal{A}(1^\lambda, c) = 1] - \Pr_{c \leftarrow \mathcal{H}_{\text{conf}_\lambda}} [\mathcal{A}(1^\lambda, c) = 1] \right| \leq \text{negl}(\lambda).$$

*Proof of Lemma. 4.* We prove this with help of a hybrid argument. To keep the proof similar to that of Lemma 3, we start with hybrid  $\mathcal{H}_{\text{conf}'}$  and end with hybrid  $\mathcal{H}_{\text{conf}}$ . □

$\mathcal{H}_{\text{conf}'}$ : The garbling step circuit is in sim mode.

$\mathcal{H}_1$ : Same as  $\mathcal{H}_1$  in Lemma 3.

$\mathcal{H}_2$ : Same as  $\mathcal{H}_2$  in Lemma 3.

$\mathcal{H}_3$ : Same as  $\mathcal{H}_3$  in Lemma 3.

$\mathcal{H}_4$ : Instead of hardwiring the PPRF key  $k$  into  $\text{Sim}_{\text{GarbleSC}}$ , we hardwire the key  $k\{i^*\} \leftarrow \text{PPRF.Punc}(s, I_w)$ , where  $I_w$  is the position of the writing head of the Turing Machine at step  $i^*$ . We additionally hardwire the labels and key that are output by  $\text{PPRF.Eval}(k, I_w)$  into  $\text{Sim}_{\text{GarbleSC}}$ .

$$k_{i^*} \leftarrow \text{PPRF.Eval}(k, I_w)$$

To be able to use the security of  $\text{iO}$ , the size of  $\text{GarbleSC}$  is padded to be the same size as  $\text{Sim}_{\text{GarbleSC}}$ .

*Claim* ( $\mathcal{H}_3 \rightarrow \mathcal{H}_4$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_3$  and  $\mathcal{H}_4$  is:

$$\text{Adv}_{\mathcal{A}_4}^{\mathcal{H}_3 \rightarrow \mathcal{H}_4} \leq \text{Adv}_{\text{iO}, \mathcal{B}_4}^{\text{iO-sec}}.$$

$\mathcal{H}_3 \rightarrow \mathcal{H}_4$ . The proof follows by a reduction to the security of the obfuscator, since the two circuits are functionally equivalent.

$\mathcal{H}_5$ : As opposed to using the key output by  $\text{PPRF.Eval}(k, I_w)$ , we sample a string  $u$  from the uniform distribution  $U_\lambda$ .

*Claim* ( $\mathcal{H}_4 \rightarrow \mathcal{H}_5$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_4$  and  $\mathcal{H}_5$  is:

$$\text{Adv}_{\mathcal{A}_5}^{\mathcal{H}_4 \rightarrow \mathcal{H}_5} \leq \text{Adv}_{\text{PPRF}, \mathcal{B}_5}^{\text{PPRF-rand}}.$$

$\mathcal{H}_4 \rightarrow \mathcal{H}_5$ . Follows by the pseudorandomness of the puncturable PRF.

$\mathcal{H}_6$ : We hardwire the output of

$$\text{GC.Garble} \left( 1^\lambda, \text{SC}_{i^*}^{\text{sim}} [\text{crs}, k_{i^*}, k_{i^*+1}, \text{lbs}', \text{lbs}'_A, \text{lbs}'_B], (\text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B; R) \right)$$

into  $\text{Sim}_{\text{GarbleSC}}$ .

*Claim* ( $\mathcal{H}_5 \rightarrow \mathcal{H}_6$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_5$  and  $\mathcal{H}_6$  is:

$$\text{Adv}_{\mathcal{A}_6}^{\mathcal{H}_5 \rightarrow \mathcal{H}_6} \leq \text{Adv}_{\text{iO}, \mathcal{C}_6}^{\text{iO-sec}}.$$

$\mathcal{H}_5 \rightarrow \mathcal{H}_6$ . The proof relies on the security of the indistinguishability obfuscator  $\text{iO}$  to be able to hardwire the output of the garbling scheme. The reduction  $\mathcal{C}_6$  gets a bit  $b$  from the adversary  $\mathcal{A}_6$ , where  $b = 0$  if the obfuscated circuit is as described in  $\mathcal{H}_5$  and  $b = 1$  if the obfuscated circuit is as described in  $\mathcal{H}_6$ . If  $\mathcal{A}_6$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_6$  wins the  $\text{iO-sec}$  game with greater than  $\epsilon$  probability.  $\square$

$\mathcal{H}_7$ : We simulate the garbling step circuit, as

$$\text{GC} \leftarrow \text{Sim}_{\text{GC}} \left( 1^\lambda, 1^{|\text{SC}_{i^*}^{\text{sim}}|}, \text{out}, (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right),$$

where  $\text{out} \leftarrow \text{SC}_{i^*}^{\text{sim}} [\text{crs}, k_{i^*}, k_{i^*+1}, \text{lbs}'_{\text{st}}, \text{lbs}'_A, \text{lbs}'_B] \left( \text{st}^{(i^*+1)}, z_A^{(i^*+1)}, z_B^{(i^*+1)} \right)$ , and  $(\text{lbs}'_{\text{st}} \parallel \text{lbs}'_A \parallel \text{lbs}'_B; R') \leftarrow \text{PPRF.Eval}(s\{i^*\}, i^* + 1)$ . Recall that  $\text{st}^{(i^*+1)}$ ,  $z_A^{(i^*+1)}$ , and  $z_B^{(i^*+1)}$  denote the state of the Turing machine  $\text{M}$ ; the digest  $\text{d}_A$  and the input block  $x_A$ ; as well as the digest  $\text{d}_B$  and the encrypted input block  $x_B$ , respectively, each at step  $i^* + 1$  of the computation.

*Claim* ( $\mathcal{H}_6 \rightarrow \mathcal{H}_7$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_6$  and  $\mathcal{H}_7$  is:

$$\text{Adv}_{\mathcal{A}_7}^{\mathcal{H}_6 \rightarrow \mathcal{H}_7} \leq \text{Adv}_{\text{GC}, \mathcal{B}_7}^{\text{GC-sec}}.$$



$\mathcal{H}_6 \rightarrow \mathcal{H}_7$ . The proof relies on the selective simulation security of the garbling scheme GC to be able to simulate the garbling step circuit. The reduction  $\mathcal{B}_7$  gets a bit  $b$  from the adversary  $\mathcal{A}_7$ , where  $b = 0$  if  $\mathcal{A}_7$  identified

$$\left\{ \text{GC.Garble} \left( 1^\lambda, \text{SC}_{i^*}^{\text{sim}} [\text{crs}, k_{i^*}, k_{i^*+1}, \text{lbs}'_{\text{st}}, \text{lbs}'_A, \text{lbs}'_B], \right. \right. \\ \left. \left. (\text{lbs}_{\text{st}} \parallel \text{lbs}_A \parallel \text{lbs}_B; R) \right), (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right\}$$

and  $b = 1$  if  $\mathcal{A}_7$  identified

$$\left\{ \text{Sim}_{\text{GC}} \left( 1^\lambda, 1^{|\text{SC}_{i^*}^{\text{sim}}|}, \text{out}, (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right), (\text{L}_{\text{st}} \parallel \text{L}_A \parallel \text{L}_B; R) \right\}.$$

If  $\mathcal{A}_7$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_7$  wins the GC-sec game with greater than  $\epsilon$  probability. □

$\mathcal{H}_8$ : Same as  $\mathcal{H}_6$  in Lemma 3.

$\mathcal{H}_9$ : Same as  $\mathcal{H}_7$  in Lemma 3.

$\mathcal{H}_{10}$ : Instead of computing the state  $\text{st}'$ , the write location  $I_w$ , the read locations  $I_r$  and  $J_r$  using  $\text{HeadPos}(i)$ , as well as computing  $y'_B$  as  $\text{Sym.Enc}(k_{i^*+1}, 0)$ ; we compute the output of  $\text{C}_M$ , and  $y'_B$  as  $\text{Sym.Enc}(k_{i^*+1}, y_B)$ .

*Claim* ( $\mathcal{H}_9 \rightarrow \mathcal{H}_{10}$ ). The advantage of any PPT adversary in distinguishing between  $\mathcal{H}_9$  and  $\mathcal{H}_{10}$  is:

$$\text{Adv}_{\mathcal{A}_{10}}^{\mathcal{H}_9 \rightarrow \mathcal{H}_{10}} \leq \text{Adv}_{\Pi, \mathcal{B}_{10}}.$$

$\mathcal{H}_9 \rightarrow \mathcal{H}_{10}$ . The proof relies on the chosen plaintext attack security of the symmetric encryption scheme  $\Pi$  to be able to switch from encrypting 0 to  $y_B$ . We can do this, since the constraints of rule  $\mathfrak{B}$  ensure that the next circuit is in *sim* mode and therefore the key  $k_{i^*+1}$  is not present in the view of the distinguisher. The reduction  $\mathcal{C}_{10}$  gets a bit  $b$  from the adversary  $\mathcal{A}_{10}$ , where  $b = 0$  if the plaintext is 0, and  $b = 1$  if the plaintext is  $y_B$ . If  $\mathcal{A}_{10}$  wins the game with advantage  $\epsilon$ ,  $\mathcal{B}_{10}$  wins the symmetric encryption game with greater than  $\epsilon$  probability. □

$\mathcal{H}_{11} - \mathcal{H}_{13}$ : Finally, we revert the changes made in  $\mathcal{H}_1 - \mathcal{H}_3$ . Here, the indistinguishability between  $\mathcal{H}_{11} - \mathcal{H}_{13}$  follows analogously to that of  $\mathcal{H}_1 - \mathcal{H}_3$ .

$\mathcal{H}_{\text{conf}}$ : The garbling step circuit is in dummy mode.

This concludes the proof of Lemma 4. ■

*Proof of Theorem 3.* The sequence of hybrids shown in the proof of Lemma 3 and Lemma 4 are reversible, and imply an inverse of rule  $\mathfrak{A}$  and rule  $\mathfrak{B}$ . Thus, the proof of Theorem 3 follows directly from the proofs of Lemma 3 and Lemma 4. ■

### 3.4 Removing the Output Dependency

We note that whilst our construction Fig. 4 outputs only one bit, a generic transformation can be used to output multiple bits. Depending on the security definition that we want to achieve, there are two generic ways to carry out such a transformation.

**Simulation Security.** If we insist on simulation security (which is the same definition achieved by the protocol in Fig. 4) we can simply hash the circuit  $\Phi$  as  $d \leftarrow \text{LFE.Hash}(\text{crs}, \Phi)$ , where  $\Phi$  takes as input an  $m_B$  and an index  $i$  and returns the  $i$ -th output bit of  $\mathbf{C}(x)_i$ . Then, for all output bits we let the sender compute

$$c := (c_1 \leftarrow \text{LFE.Enc}(\text{crs}, d, (x, 1)), \dots, c_{|y|} \leftarrow \text{LFE.Enc}(\text{crs}, d, (x, |y|)))$$

where  $|y|$  denotes the output size. The receiver can then recover the output bit-by-bit. Security follows from a standard hybrid argument.

**Indistinguishability.** If we relax the requirements to indistinguishability-based security, then it becomes possible to remove the output dependency entirely. Specifically, we require that  $\text{LFE.Enc}(\text{crs}, d, x)$  and  $\text{LFE.Enc}(\text{crs}, d, \bar{x})$  are computationally indistinguishable, for pairs  $(x, \bar{x})$  such that  $\mathbf{C}(x) = \mathbf{C}(\bar{x})$ .

Our scheme proceeds as described above except that the sender does not explicitly compute the ciphertexts  $(c_1, \dots, c_{|y|})$ , instead the sender obfuscates a circuit that given an index  $i \in \{1, \dots, |y|\}$  returns

$$\text{LFE.Enc}(\text{crs}, d, (x, i); \text{PPRF.Eval}(k, i))$$

where  $k$  is the key of a puncturable PRF. To compute the output, the receiver evaluates the obfuscated circuit on all possible indices to recover  $(c_1, \dots, c_{|y|})$ , then she applies the  $\text{LFE.Dec}$  algorithm to recover the output bit-by-bit. Observe that now the size of the ciphertext depends on  $|y|$  only logarithmically.

In terms of security, we can show indistinguishability by defining  $(|y|+1)$ -many intermediate distributions, where in the  $i^*$ -th distribution  $\mathcal{H}_{i^*}$  we obfuscate the circuit that given an index  $i \in \{1, \dots, |y|\}$  returns

$$\begin{aligned} &\text{LFE.Enc}(\text{crs}, d, (\bar{x}, i); \text{PPRF.Eval}(k, i)), && \text{if } i < i^* \\ &\text{LFE.Enc}(\text{crs}, d, (x, i); \text{PPRF.Eval}(k, i)), && \text{otherwise.} \end{aligned}$$

Note that  $\mathcal{H}_0$  is functionally equivalent to the original obfuscated circuit, whereas  $\mathcal{H}_{|y|+1}$  is functionally equivalent to the encryption of  $\bar{x}$ . Thus, it suffices to show that  $\mathcal{H}_{i^*}$  and  $\mathcal{H}_{i^*+1}$  are computationally indistinguishable. This is done with help of a five-steps argument:

- First we puncture the PRF key at point  $i^*$ , and indistinguishability follows from the security of  $\text{iO}$ .
- We switch  $\text{PPRF.Eval}(k, i^*)$  with a uniform string  $u$ , which is indistinguishable by the security of the puncturable PRF.

- We hardwire the output of  $c^* \leftarrow \text{LFE.Enc}(\text{crs}, d, (x, i^*); u)$  in the obfuscated circuit. Again, indistinguishability follows from the security of  $iO$ .
- We set  $c^* \leftarrow \text{LFE.Enc}(\text{crs}, d, (\bar{x}, i^*); u)$ . Indistinguishability follows from the security of LFE.
- We undo the modifications done by the first three steps.

Note that the first distribution is identical to  $\mathcal{H}_{i^*}$ , whereas the latter is identical to  $\mathcal{H}_{i^*+1}$ .

## References

- [ACC+16] Ananth, P., Chen, Y.-C., Chung, K.-M., Lin, H., Lin, W.-K.: Delegating RAM computations with adaptive soundness and privacy. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 3–30. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53644-5\\_1](https://doi.org/10.1007/978-3-662-53644-5_1)
- [AFS19] Ananth, P., Fan, X., Shi, E.: Towards attribute-based encryption for RAMs from LWE: sub-linear decryption, and more. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 112–141. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34578-5\\_5](https://doi.org/10.1007/978-3-030-34578-5_5)
- [AJS17] Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation for turing machines: constant overhead and amortization. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 252–279. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63715-0\\_9](https://doi.org/10.1007/978-3-319-63715-0_9)
- [AR21] Agrawal, S., Rosie, R.: Adaptively secure laconic function evaluation for NC1. E-prints/Working papers: ORBilu, 2021. <https://orbilu.uni.lu/handle/10993/46493>
- [BDGM20] Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Factoring and pairings are not necessary for  $iO$ : Circular-secure LWE suffices. Cryptology ePrint Archive, Report 2020/1024 (2020). <https://eprint.iacr.org/2020/1024>
- [BFK+19] Badrinarayanan, S., Fernando, R., Koppula, V., Sahai, A., Waters, B.: Output compression, MPC, and  $iO$  for turing machines. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 342–370. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34578-5\\_13](https://doi.org/10.1007/978-3-030-34578-5_13)
- [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](https://doi.org/10.1007/3-540-44647-8_1)
- [BGL+15] Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: Servedio, R.A., Rubinfeld, R.M editors, 47th Annual ACM Symposium on Theory of Computing, pp. 439–448, Portland, OR, USA, June 14–17, ACM Press (2015)
- [BP15] Boyle, E., Pass, R.: Limits of Extractability Assumptions with Distributional Auxiliary Input. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 236–261. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48800-3\\_10](https://doi.org/10.1007/978-3-662-48800-3_10)
- [BSW16] Bellare, M., Stepanovs, I., Waters, B.: New Negative Results on Differing-Inputs Obfuscation. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 792–821. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49896-5\\_28](https://doi.org/10.1007/978-3-662-49896-5_28)

- [CCC+16] Chen, Y.C., Chow, S.S., Chung, K.M., Lai, R.W., Lin, W.K., Zhou, H.S.: Cryptography for parallel RAM from indistinguishability obfuscation. In Madhu Sudan, editor, *ITCS 2016: 7th Conference on Innovations in Theoretical Computer Science*, pp. 179–190, Cambridge, MA, USA, January 14–16, Association for Computing Machinery (2016)
- [CCHR15] Canetti, R., Chen, Y., Holmgren, J., Raykova, M.: Succinct adaptive garbled RAM. *Cryptology ePrint Archive*, Report 2015/1074 (2015). <https://eprint.iacr.org/2015/1074>
- [CDG+17] Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) *CRYPTO 2017*. LNCS, vol. 10402, pp. 33–65. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63715-0\\_2](https://doi.org/10.1007/978-3-319-63715-0_2)
- [CH16] Canetti, R., Holmgren, J.: Fully succinct garbled RAM. In: Sudan, M., editor, *ITCS 2016: 7th Conference on Innovations in Theoretical Computer Science*, pp. 169–178, Cambridge, MA, USA, January 14–16, 2016. Association for Computing Machinery (2016)
- [GGHW14] Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica* **79**(4), 1353–1373 (2017). <https://doi.org/10.1007/s00453-017-0276-6>
- [GGSW13] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J., editors, *45th Annual ACM Symposium on Theory of Computing*, pp. 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press (2013)
- [GKP+13] 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](https://doi.org/10.1007/978-3-642-40084-1_30)
- [GP20] Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. *Cryptology ePrint Archive*, Report 2020/1010 (2020). <https://eprint.iacr.org/2020/1010>
- [GPSZ17] Garg, S., Pandey, O., Srinivasan, A., Zhandry, M.: Breaking the sub-exponential barrier in obfustopia. In: Coron, J.-S., Nielsen, J.B. (eds.) *EUROCRYPT 2017*. LNCS, vol. 10212, pp. 156–181. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_6](https://doi.org/10.1007/978-3-319-56617-7_6)
- [GS18] Garg, S., Srinivasan, A.: A simple construction of io for turing machines. In: Beimel, A., Dziembowski, S. (eds.) *TCC 2018*. LNCS, vol. 11240, pp. 425–454. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03810-6\\_16](https://doi.org/10.1007/978-3-030-03810-6_16)
- [HW15] Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T., editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pp. 163–172, Rehovot, Israel, January 11–13, 2015. Association for Computing Machinery (2015)
- [JLS20] 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>

- [KLW15] Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: Rocco A. Servedio and Ronitt Rubinfeld, editors, 47th Annual ACM Symposium on Theory of Computing, pp. 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press (2015)
- [KNYY19] Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Exploring constructions of compact NIZKs from various assumptions. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 639–669. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_21](https://doi.org/10.1007/978-3-030-26954-8_21)
- [NRS21] Naccache, D., Rosie, R., Spignoli, L.: Post-quantum secure lfe for L/poly with smaller parameters. E-prints/Working papers: ORBilu, (2021). <https://hdl.handle.net/10993/46725>
- [OPWW15] 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. LNCS, vol. 9452, pp. 121–145. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48797-6\\_6](https://doi.org/10.1007/978-3-662-48797-6_6)
- [PCFT20] Pang, B., Chen, L., Fan, X., Tang, Q.: Multi-input laconic function evaluation. In: Liu, J.K., Cui, H. (eds.) ACISP 2020. LNCS, vol. 12248, pp. 369–388. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-55304-3\\_19](https://doi.org/10.1007/978-3-030-55304-3_19)
- [PF79] Pippenger, N., Fischer, M.J.: Relations among complexity measures. J. ACM, **26**(2), 361–381 (1979)
- [QWW18] Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: Thorup, M., editor, 59th Annual Symposium on Foundations of Computer Science, pp. 859–870, Paris, France, October 7–9, 2018. IEEE Computer Society Press (2018)
- [WW21] Wee, Hoeteck, Wichs, Daniel: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, Anne, Standaert, François-Xavier. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 127–156. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77883-5\\_5](https://doi.org/10.1007/978-3-030-77883-5_5)
- [Yao82] Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press
- [Yao86] Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In 27th Annual Symposium on Foundations of Computer Science, pp. 162–167, Toronto, Ontario, Canada, October 27–29 (1986). IEEE Computer Society Press



# A Map of Witness Maps: New Definitions and Connections

Suvradip Chakraborty<sup>1(✉)</sup>, Manoj Prabhakaran<sup>2</sup>, and Daniel Wichs<sup>3,4</sup>

<sup>1</sup> Visa Research, Palo Alto, USA  
suvradip1111@gmail.com

<sup>2</sup> Department of Computer Science, IIT Bombay, Mumbai, India  
mp@cse.iitb.ac.in

<sup>3</sup> Northeastern University, Boston, USA  
wichs@ccs.neu.edu

<sup>4</sup> NTT Research, Palo Alto, USA

**Abstract.** A *witness map* deterministically maps a witness  $w$  of some NP statement  $x$  into computationally sound proof that  $x$  is true, with respect to a public common reference string (CRS). In other words, it is a deterministic, non-interactive, computationally sound proof system in the CRS model. A *unique witness map* (UWM) ensures that for any fixed statement  $x$ , the witness map should output the same *unique* proof for  $x$ , no matter what witness  $w$  it is applied to. More generally a *compact witness map* (CWM) can only output one of at most  $2^\alpha$  proofs for any given statement  $x$ , where  $\alpha$  is some compactness parameter. Such compact/unique witness maps were proposed recently by Chakraborty, Prabhakaran and Wichs (PKC '20) as a tool for building tamper-resilient signatures, who showed how to construct UWMs from indistinguishability obfuscation (iO). In this work, we study CWMs and UWMs as primitives of independent interest and present a number of interesting connections to various notions in cryptography.

- First, we show that UWMs lie somewhere between witness PRFs (Zhandry; TCC '16) and iO – they imply the former and are implied by the latter. In particular, we show that a relaxation of UWMs to the “designated verifier (dv-UWM)” setting is *equivalent* to witness PRFs. Moreover, we consider two flavors of such dv-UWMs, which correspond to two flavors of witness PRFs previously considered in the literature, and show that they are all in fact equivalent to each other in terms of feasibility.
- Next, we consider CWMs that are extremely compact, with  $\alpha = O(\log \kappa)$ , where  $\kappa$  is the security parameter. We show that such CWMs imply *pseudo-UWMs* where the witness map is allowed to be *pseudo-deterministic* – i.e., for every true statement  $x$ , there is a unique proof such that, on any witness  $w$ , the witness map outputs

---

S. Chakraborty—Work done while the author was at ETH Zurich.

M. Prabhakaran—Supported by the IITB Trust Lab.

D. Wichs—Research supported by NSF grant CNS-1750795, CNS-2055510, the Alfred

P. Sloan Research Fellowship.

© International Association for Cryptologic Research 2023

A. Boldyreva and V. Kolesnikov (Eds.): PKC 2023, LNCS 13941, pp. 635–662, 2023.

[https://doi.org/10.1007/978-3-031-31371-4\\_22](https://doi.org/10.1007/978-3-031-31371-4_22)

this proof with  $1 - 1/p(\lambda)$  probability, for a polynomial  $p$  that we can set arbitrarily large.

- Lastly, we consider CWMs that are mildly compact, with  $\alpha = p(\lambda)$  for some a-priori fixed polynomial  $p$ , independent of the length of the statement  $x$  or witness  $w$ . Such CWMs are implied by succinct non-interactive arguments (SNARGs). We show that such CWMs imply NIZKs, and therefore lie somewhere between NIZKs and SNARGs.

## 1 Introduction

When several mathematicians prove the same theorem, it is unlikely that they would all write down the exact same proof. Similarly, in the context of NP, a true statement (e.g., that some graph is 3-colorable) will often have many different proofs/witnesses (e.g., 3-colorings of the vertices). Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?

This question was studied extensively in complexity theory, where the class of languages with unique proofs is known as UP [15]. It is believed to be unlikely that  $\text{NP} = \text{UP}$ , meaning that we do not believe that all NP languages have unique proof systems, and there are several results that separate the two classes relative to oracles [1, 2, 13]. Recently, the work of Chakraborty, Prabhakaran and Wichs [4] proposed unique proof systems with *computational soundness* (aka arguments). They defined the notion of a *unique witness map* (UWM) in the common reference string (CRS) model. This is a deterministic polynomial-time map that takes as input an NP statement  $x$  and some arbitrary witness  $w$  for  $x$  (and the CRS) and maps them to a unique proof  $w^*$  for  $x$ . Any other witness  $w'$  for  $x$  is mapped to the same unique proof  $w^*$ . There is also a polynomial time verifier that checks whether  $w^*$  is a good proof of the statement  $x$ . The computational soundness guarantee ensures that no polynomial time adversary can cause the verifier to accept a proof of a false statement  $x$ , except with negligible probability over the choice of the CRS. In other words, a UWM is a deterministic non-interactive computationally sound proof (aka argument) system with unique proofs. More generally, [4] also considered a relaxation of UWMs to *compact witness maps* (CWMs) where the number of possible proofs  $w^*$  that the map can output for any given statement  $x$  is bounded by  $2^\alpha$ , for some *compactness parameter*  $\alpha$ . UWMs then naturally correspond to CWMs with  $\alpha = 0$ .

It is worth noting that UWMs/CWMs only impose a restriction on the number of proofs  $w^*$  that the prover outputs, but not on the number of proofs that the verifier accepts for a given statement  $x$ . It may be the case that we have a UWM where the prover outputs a unique proof  $w^*$  for a given statement  $x$ , but there are exponentially many alternate proofs that the verifier would accept as well. We also note that UWMs are easily seen to be a special case of a *witness-indistinguishable* proof system, since all witnesses are mapped to the same unique proof.

The work of [4] showed how to construct UWMs from indistinguishability obfuscation (iO) and one-way functions, closely following the construction of NIZKs due to [14]. They also showed that UWMs imply witness encryption. As their main result, they gave an application of UWMs to the problem of leakage and tamper-resilient signatures with a deterministic signer. However, not much else was known about UWMs/CWMs and how they relate to other notions in cryptography.

## 1.1 Our Results

In this work, we undertake the thorough study of UWMs/CWMs as primitives of interest in their own right. We provide a number of novel results to better understand these notions and discover surprising connections between UWM/CWM and other cryptographic objects of interest. Interestingly, we show that (quantitative) compression factor affects the (qualitative) cryptographic power, leading to a hierarchy of “worlds”, depending on whether all of NP has  $\alpha$ -CWM for, say,  $\alpha = 0, O(1), \log n, n^c$  ( $c < 1$ ), somewhat akin to Impagliazzo’s worlds.

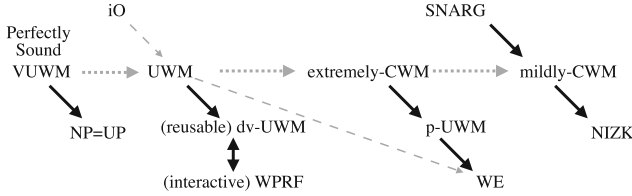
The study of UWMs/CWMs can be seen as part of a broader context of complexity theoretic study within cryptography, whose aim is to understand connections between primitives and their relative power. We also view the study of UWMs/CWMs as adding to the understanding of “functional compression” as a fundamental cryptographic feature. For example, functional compression plays a central role in obfuscation, where we can define variants (e.g.,  $XiO$  vs  $iO$  [11]) depending on the level of compression provided. And (perhaps a bit further off), the complexity of computing Kolmogorov complexity, which is also about functional compression, is deeply related to the existence of one-way functions [12].

We now discuss each of our results, relating CWMs with various levels of compression to other cryptographic objects and to each other.

**Relating UWMs and Witness PRFs.** At its most compact end, witness maps take the form of UWMs. We show several results tightly relating flavors of UWMs and flavors of witness PRFs.

First, we show that UWMs imply witness PRFs [17], which lie somewhere between witness encryption and iO, but are believed to be strictly stronger than witness encryption. In particular, they were shown to imply multi-party key exchange without trusted setup, polynomially-many hardcore bits for any one-way function, and several other applications that are otherwise only known from iO, but not from witness encryption.





**Fig. 1.** A summary of the implications established (under standard cryptographic assumptions). The dotted lines correspond to trivial implications, and the dashed line are results from [4].

In a witness PRF, just like a standard PRF, there is a secret function key  $fk$  that allows the holder to evaluate the function on any input  $x$ . However, there is also a public evaluation key  $ek$ , that allows one to evaluate the function on any input  $x$  belonging to some NP language  $L$ , provided the evaluator also has the corresponding witness  $w$ . The basic security notion says that for any  $x \notin L$ , the output of the function looks uniform even given the public evaluation key. A stronger *interactive* security variant says that the above should hold even if the adversary can query the function on arbitrary other inputs  $x \notin L$ . It is trivial to construct witness encryption from witness PRFs (with basic security), but the other direction is not known.

We show that UWM and one-way functions imply witness PRFs. In fact, we show that witness PRFs are equivalent to a weaker form of designated-verifier UWMs (dv-UWMs), where the public CRS is generated together with a secret verification key needed to verify proofs. In this case we can define two flavors of soundness: A basic soundness guarantee when the adversarial prover does not get any information about the secret verification key, beyond seeing the CRS, and a *reusable soundness* guarantee for dv-UWMs, where the adversarial prover can make verification queries to check whether purported proofs for various (true or false) statements  $x$  would be accepted using the verification key. We show that all four notions are equivalent in terms of feasibility (assuming one-way functions): witness PRFs with interactive security imply reusable dv-UWMs, which imply basic dv-UWMs, which imply basic witness PRFs, which then imply interactive witness PRFs. In particular, the last result shows that it is possible to generically upgrade witness PRFs with basic security to interactive security.

The above results place UWMs on the map somewhere between witness PRFs (which are equivalent to dv-UWMs) and iO. We also believe that UWMs are likely stronger than witness PRFs and dv-UWMs, mainly since we do not know of any way to generically go from the designated verifier setting to public verifiability. Moreover, we show that UWMs imply non-interactive zero-knowledge (NIZKs) with a deterministic prover, which are currently only known from iO, but not from witness PRFs.

**Extreme Compactness Implies Pseudo-Uniqueness.** Next, we consider CWMs with “extreme compactness”  $\alpha = O(\log \kappa)$  for security parameter  $\kappa$ . In

other words, while we do not require the proofs to be unique, we require that the witness map can produce at most  $2^\alpha = \text{poly}(\kappa)$  many possible proofs for each statement  $x \in L$ . We show that such extreme compactness is almost as good as uniqueness. In particular, we show that one can generically transform an extremely compact CWM into a pseudo-unique witness map (pseudo-UWM), where the pseudo-uniqueness property says the following. For any statement  $x \in L$  and any two witnesses  $w_1, w_2$  for  $x$ , both witnesses will map to the same “pseudo-unique” proof  $w^*$  with high  $1 - 1/p(\kappa)$  probability over the choice of the CRS, where we can choose  $p(\kappa)$  to be an arbitrarily large polynomial.

We note that a pseudo-UWM remains a powerful primitive. It can be used instead of UWM in applications where the “error” (i.e., non-uniqueness) can be “corrected.” In particular, it implies witness encryption. Indeed, in the construction of witness encryption using a UWM (from [4], or alternately, from a WPRF which is in turn constructed from a UWM as shown here), if we simply replace the UWM with a pseudo-UWM, it results in a small decryption error probability. This error probability can be made exponentially low by repeating the encryption process multiple times using independent keys and randomness, and during decryption, outputting the majority.

To show that extremely compact WMs imply pseudo-UWM, we solve an abstract problem of potentially independent interest that we refer to as *pseudo-deterministic sampling*. Consider a sampler that has oracle access to some arbitrary distribution  $\mathcal{D}$  whose support has polynomial size. The sampler can call the oracle polynomially many times and each call outputs a fresh random sample  $x \leftarrow \mathcal{D}$ . At the end, the sampler has to output some value  $x^*$  in the support of the distribution  $\mathcal{D}$ . Moreover, we want the sample  $x^*$  to be unique; if we run the sampler twice, with the oracle producing random/independent samples from  $\mathcal{D}$  in each run, the sampler should output the same value  $x^*$  in both executions with high  $1 - 1/p(\kappa)$  probability. This guarantee is similar to pseudo-deterministic algorithms [6, 8], which are randomized algorithms that nevertheless output a unique value independent of their randomness with high probability. We show how to solve the pseudo-deterministic sampling problem in the CRS model. The sample  $x^*$  that the sampler outputs may depend on the CRS but, with high probability, should be the same for every execution of the sampler with the given CRS, no matter what samples it receives from its oracle.

**Mild Compactness Implies NIZKs.** We then turn our attention to CWMs with “mild compactness” where  $\alpha = p(\kappa)$  for some fixed polynomial  $p$ , independent of the statement size  $|x|$  or the witness size  $|w|$ . Such CWMs are implied by succinct non-interactive arguments (SNARGs) for NP, which are computationally sound proofs where the proof size is bounded by some fixed polynomial  $p(\kappa)$ , and independent of  $|x|$  or  $|w|$ . The mild compactness of CWMs can be seen as a relaxation of the succinctness requirement for SNARGs, where the latter requires the proof to have small size  $p(\kappa)$ , while the former only requires the number of possible proofs that the prover outputs to be bounded by  $2^{p(\kappa)}$  but allows the size of the proofs to be arbitrarily large. Although mildly compact

CWMs are weaker than SNARGs, we show that they nevertheless imply non-interactive zero knowledge (NIZK) proofs. We generalize the recent work of [10], who showed how to construct NIZKs from SNARGs, by showing that the same result holds if we replace SNARGs by mildly compact CWMs. The above shows that mildly compact CWMs lie somewhere between NIZKs and SNARGs.

**UWMs with Statistical Soundness and UP.** Lastly, we ask whether we can get UWMs with statistical/perfect soundness. This appears highly unlikely since it would imply a construction of witness PRFs (and hence witness encryption) from one-way functions. But can we rule out this possibility under some well-studied complexity assumption? Interestingly, we do not know the answer to this question. Intuitively, we'd like to say that perfectly sound UWMs for NP would imply  $\text{NP} = \text{UP}$ , where UP is the class of languages where every statement  $x$  has a unique witness  $w^*$ . However, a perfectly sound UWM only guarantees that the prover outputs a unique proof and that the verifier never accepts a proof for a false statement, but it may still be possible for the verifier to accept many possible proofs besides the one that the prover outputs. We define the stronger notion of verifier-unique witness maps (VUWM) where we also guarantee that the verifier only accepts a unique proof  $w^*$  for each  $x$ , and show that perfectly sound VUWMs for NP imply  $\text{NP} = \text{UP}$ .

## 1.2 Technical Overview

### 1.2.1 $\text{dv-UWM}$ Is Equivalent to Witness PRFs

To show the equivalence between  $\text{dv-UWM}$  and witness PRF  $\text{wPRF}$ , we first show that  $\text{wPRF}$  implies  $\text{dv-UWM}$ .

**Witness PRF Implies  $\text{dv-UWM}$ :** This direction is rather straightforward and follows from the definition of  $\text{wPRF}$ . In particular, the  $\text{dv-UWM}$  proof  $w^*$  is computed by running the public evaluation algorithm using the evaluation key  $\text{ek}$ . The verification algorithm of  $\text{dv-UWM}$  is obtained by running the secret evaluation algorithm of  $\text{wPRF}$  using the secret function key  $\text{fk}$  and checking if the proof  $w^*$  is equal to the output of this algorithm. The correctness of the construction follows from the fact that the values computed in both the modes of  $\text{wPRF}$  are equal. Uniqueness is guaranteed since the private evaluation algorithm does not depend on the witness  $w$  and deterministically maps  $x$  to a unique output value. Finally, the soundness of  $\text{dv-UWM}$  follows from the interactive security of  $\text{wPRF}$ .

**$\text{dv-UWM}$  Implies Witness PRF:** We show this result in two steps – (i) First, we show that a construction of non-interactive witness PRF for  $\text{NP}$  from any non-reusably sound  $\text{dv-UWM}$  for  $\text{NP}$ , (ii) next, we show a generic transformation from any non-interactive witness PRF for  $\text{NP}$  to an interactive witness PRF for  $\text{NP}$  additionally using one way function.

*Non-reusably sound UWM Implies Non-interactive Witness PRF:* To construct a non-interactive  $\text{wPRF}$  from a (non-reusable)  $\text{dv-UWM}$ , the key generation algorithm  $\text{wPRF.Gen}$  of  $\text{wPRF}$  samples a random seed  $z$  for a (length-doubling) pseudorandom generator  $G$  and sets  $y = G(z)$ . It then runs the setup algorithm of

DV-UWM, i.e., `dv.setup` to obtain  $((K, VK))$ . It then sets the evaluation key as  $ek = (K, VK, y)$  and the function key as  $fk = z$ . To compute the function  $F(fk, \cdot)$  on input  $x \in L$ , the evaluator uses DV-UWM to get a representative witness  $w^*$  for the statement  $\hat{x}$  stating that “either  $x$  is true or  $y$  is pseudorandom”, using  $z$  as the witness. It then outputs a hardcore bit (e.g., the Goldreich-Levin (GL) hardcore bit) of  $w^*$  as the pseudorandom bit  $b$ . In the public evaluation mode, on input  $(x, w)$  the algorithm `wPRF.Eval` uses the UWM to map the witness  $w$  for  $x$  into the unique witness  $w^*$  for the statement  $\hat{x}$ . It can then compute the pseudorandom bit  $b$  using the GL predicate. Intuitively, if an adversary can break wPRF security, then it can distinguish the bit 0 and 1 with non-negligible probability even if  $x$  is a false statement. This means that, using GL decoding, it can compute the correct value  $w^*$  given  $y$  with non-negligible probability. Furthermore this value  $w^*$  is a valid representative witness for the statement  $\hat{x}$ . Since the adversary cannot break the PRG, it must also compute a valid representative witness for  $\hat{x}$  if we switch  $y$  to false. But this contradicts the soundness of dv-UWM.

*Generic Transformation from Non-interactive to Interactive Witness PRF:* To construct interactive witness PRF from non-interactive witness PRF (nl-wPRF) we need to carefully define the relation for the underlying nl-wPRF. In particular, the key generation algorithm of the interactive witness PRF wPRF runs the key generation algorithm of the non-interactive witness PRF nl-wPRF to obtain  $(\hat{ek}, \hat{fk})$ . It then uses a statistically binding commitment scheme to commit to a message  $\mathbf{0}$  (using randomness  $r$ ) such that  $\mathbf{0}$  is not a valid statement of the underlying NP relation  $\mathcal{R}$  for wPRF to obtain commitment  $c$ . In other words,  $\mathbf{0} \notin \mathcal{X}$ , where  $\mathcal{X}$  is the statement space of  $\mathcal{R}$ . It then sets the evaluation key as  $ek = (\hat{ek}, c)$  and the function key as  $fk = (\hat{fk}, r)$ . To compute the function  $F(fk, \cdot)$  on input  $x \in L$ , the evaluator uses nl-wPRF to get a value  $y$  for the statement  $\hat{x}$  stating that “either  $x$  is true or  $c$  is a commitment to some message  $x'$  such that  $x \neq x'$ ”. In the public evaluation mode, on input  $(x, w)$  the algorithm `wPRF.Eval` uses the public evaluation key of the underlying nl-wPRF to map the witness  $w$  for  $x$  into the value  $y$  for the statement  $\hat{x}$ . In the proof, when the adversary commits to a challenge  $x^* \notin L$ , we switch from a commitment to  $\mathbf{0}$  to a commitment  $c^*$  to  $x^*$ . Hence, we have that the statement  $\hat{x} = (x^*, c^*)$  is now false. The hiding property of the commitment allows us to make such a switch. On the other hand, for all other statement  $x_i \neq x^*$ , the statement  $(x_i, c^*)$  is still true, and hence we can simulate the queries of the wPRF adversary using the function `nl-wPRF.F( $\hat{fk}, \cdot$ )`.

### 1.2.2 Extremely Compact WM Implies Pseudo-Unique WM

As mentioned above, to construct a Pseudo-UWM from an extremely compact WM, we solve the abstract problem of *pseudo-deterministic sampling* from a distribution with polynomial-sized support. We briefly sketch our solution to the pseudo-deterministic sampling problem.

**Solving Pseudo-Deterministic Sampling.** As a first attempt, consider obtaining  $N$  samples from the distribution for a value  $N$  that is much larger

than the size of the distribution’s support, and then taking the lexicographically smallest one. This would indeed work if we could ensure that every element in the support gets sampled at least once. Unfortunately, this does not hold true for arbitrary distributions. For instance, if the lexicographically smallest element in the support has a probability  $1/N$ , then there is a constant probability for it to get sampled as well as to not get sampled. As a second attempt, one may consider using a hash function to define the lexicographic ordering to prevent the distribution from adversarially assigning such a probability to the smallest element; however, this does not help either, if a large fraction of the elements in the support have probability  $1/N$ . One may note that the difficulty here arises from (moderately) low probability elements; so to avoid such elements, we could try to pick a high probability element, which is guaranteed to occur many times in the sample. However neither picking the most frequent element (e.g., when there are multiple elements which have the maximum probability), nor picking the lexicographically smallest one from among a selected subset of frequent elements (e.g., when there are elements with probabilities that place them near the threshold used for selection) is sufficient to guarantee uniqueness. Our final solution combines ideas from all of these approaches: it obtains  $N$  samples and would choose one with the smallest hash value, but the hash is computed on *the element concatenated with a counter*. That is, if an element  $x$  occurs  $k$  times in the sample, then the hashes of all of  $x||1, x||2, \dots, x||k$  are considered. This has the effect of picking an element from among the more frequent elements, but without creating a threshold for being considered frequent. Using elementary concentration bounds we show that the probability of two executions of this process yielding different outcomes (when using a  $N$ -wise independent hash function) goes down polynomially with  $1/N$ .

**Pseudo-UWM from Pseudo-Deterministic Sampling.** To reduce pseudo-UWM to pseudo-deterministic sampling, first we need to create a distribution over proofs that remains (essentially) the same for all witnesses. We achieve this using a Non-Interactive Witness Indistinguishable proof system (NIWI). Then we map this NIWI proof using the extremely compact WM (for the relation corresponding to NIWI verification) into a polynomial-sized support. The soundness of this proof depends on the fact that for any false statement, there should not *exist* a NIWI proof that gets accepted; this is guaranteed by using a statistically sound NIWI. At this point, we have a proof system that is sound, compact, and witness-indistinguishable. Now, pseudo-deterministic sampling from this distribution would result in a pseudo-UWM.

### 1.2.3 Mildly CWM Implies NIZK

We show that CWM with compactness level  $\alpha = \text{poly}(\kappa)$  for some fixed polynomial  $\text{poly}(\cdot)$ , independent of the statement size  $|x|$  or the witness size  $|w|$  implies the existence of NIZK argument system. Our construction generalizes the recent work of [10] who constructed NIZKs from SNARG by replacing SNARG with CWM with the above compactness level. [10] shows how to compile any NIZK in the hidden-bits model (HBM) to NIZK in the CRS model using a primitive called

*hidden-bits generator with subset-dependent proofs* (SDP-HBG). Then they they show how to construct such a SDP-HBG from any SNARG and bounded-leakage weak PRF (BLR-wPRF)<sup>1</sup>. Below we sketch the main idea of the construction and the proof of [10]. We then show how to modify their construction and proof technique when using CWM instead of SNARG. A SDP-HBG consists of the following algorithms:

- $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$  generates a CRS  $\text{crs}$  where  $n$  denotes the length of hidden-bits to be generated.
- $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$  generates “hidden-bits”  $r \in \{0, 1\}^n$  and a state  $\text{st}$ .
- $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$  generates a proof  $\pi$  that certifies the sub-string  $r_I$ .
- $\text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi)$  verifies the proof  $\pi$  to ensure that the substring of  $r$  on the positions corresponding to subset  $I$  is indeed  $r_I$ .

The SDP-HBG is required to satisfy the following properties – (i) *Somewhat Computational Binding*, which requires that exists a “sparse” subset  $\mathcal{V}^{\text{crs}} \in \{0, 1\}^n$  of size much smaller than  $2^n$  such that no PPT malicious prover can generate a proof for bits that are not consistent with any element of  $\mathcal{V}^{\text{crs}}$ , (ii) *Hiding*, which requires that for any subset  $I \subseteq [n]$ , no PPT adversary given can distinguish  $r_I$  from a uniformly random string  $r'_I$ , where  $r_I$  denotes the substring of  $r$  on the positions corresponding to  $\bar{I} = [n] \setminus I$ . To construct SDP-HBG, the setup algorithm  $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$  samples  $\vec{x} = (x_1, \dots, x_n) \in \{0, 1\}^{m \times n}$  and sets  $\text{crs} = \vec{x}$ . The algorithm  $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$  derives the hidden bits  $\vec{r} = (r_1, \dots, r_n) \in \{0, 1\}^n$  as  $r_i = F_K(x_i)$ , where  $F_K : \{0, 1\}^m \rightarrow \{0, 1\}$  is a  $\lambda$ -BLR-wPRF and  $K \in \{0, 1\}^k$  for some polynomial  $k = k(\kappa, \lambda)$ , where  $\kappa$  is the security parameter. The algorithm  $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$  then uses the SNARG to generate a proof  $\pi$  for the statement that the values  $r_i$  for all  $i \in [I]$  are correctly computed, using  $K$  as the witness. The verification then consists of verifying the SNARG proof.

The somewhat computational binding property of SDP-HBG easily follows from the soundness of SNARG as long as  $k \ll n$  (where  $k$  is the size of the PRF key  $K$  and  $n$  is the length of the hidden bit string). The hiding property is easy to reduce to security of the underlying BLR-wPRF as long as  $|\pi| \leq \lambda$ . In particular, the proof  $\pi$  corresponds to the subset  $I$  which does not depend on  $x_{\bar{I}}$  (the bits of  $x$  in the positions  $[n] \setminus I$ ), and thus we can think of  $x_{\bar{I}}$  as the challenge inputs and  $\pi$  as the leakage.

*Using CWM instead of SNARG.* Our construction follows the same blueprint from [10], except that we use a  $\alpha$ -CWM for  $\alpha = \text{poly}(\kappa)$  instead of a SNARG to generate the proof  $\pi$  and use a entropic leakage-resilient weak PRF<sup>2</sup> to generate

<sup>1</sup> Informally, a  $\lambda$ -BLR-wPRF  $F_K(\cdot)$  guarantees pseudorandomness of the output of the PRF when evaluated on uniformly random inputs, even when the adversary can leak up to  $\lambda$  bits on  $K$ . [9] showed how to construct such BLR-wPRF from any OWF.

<sup>2</sup> Informally, a  $\lambda$ -entropic leakage-resilient PRF  $F_K(\cdot)$  guarantees pseudorandomness of the output of the PRF when evaluated on uniformly random inputs, even when the adversary can get  $\lambda$ -entropic leakage on  $K$ . Roughly this means that the PRF key  $K$  still has  $k - \lambda$  bits of average min-entropy, even conditioned on the leakage from  $K$ . [9] showed how to construct such entropic LR-wPRF from any OWF.

the hidden bits  $r \in \{0, 1\}^n$ , instead of a bounded leakage-resilient weak PRF. This is because, unlike SNARG the size of our CWM proofs  $\pi$  are *not* guaranteed to be succinct. However, we have the guarantee that the proof  $\pi$  is  $\alpha$ -compact, for some  $\alpha = \text{poly}(\kappa)$ , where  $\text{poly}(\kappa)$  is independent of  $n$ . This means the size of the CWM image is at most  $2^\alpha$ . Hence, as long as the underlying wPRF is resilient to  $\lambda$ -entropic leakage, where  $\alpha \leq \lambda$ , we can rely on the (entropic) leakage-resilience of ELR-wPRF to argue hiding of the SDP-HBG. We can then set the parameters appropriately to satisfy these two inequalities.

#### 1.2.4 UWM Implies Deterministic-prover NIZK

We show that UWM implies deterministic prover NIZK arguments systems (DP-NIZK), where the prover and verifier are deterministic. In fact, we can achieve *perfect* zero-knowledge property. The main idea of our construction is similar to the construction of non-interactive witness PRF from dv-UWM. In particular, the setup algorithm of DP-NIZK chooses a pseudorandom string  $y = G(z)$ , where  $G$  is a length-doubling PRG. The CRS  $\text{crs}$  of DP-NIZK consists of the CRS  $K$  of UWM and the value  $y$ . The prover of DP-NIZK on input some  $(x, w)$  in the relation runs the UWM prover to get a representative witness  $w^*$  for the statement  $\hat{x}$  stating that “either  $x$  is true or  $y$  is pseudorandom”, using  $w$  as the witness. Note that the prover is deterministic. The verification of DP-NIZK simply uses the UWM verifier. To prove the soundness of this construction, we sample  $y$  uniformly at random and hence with very high probability there does not exist any valid preimage of  $y$  with respect to  $G$ . Hence, if  $x \notin L$ , the statement  $\hat{x} = (x, y)$  is also not in the (augmented) language with overwhelming probability. The soundness of the construction now follows from the soundness of UWM. To prove zero-knowledge property, the simulator uses the trapdoor  $z$  as the witness to simulate proofs of statements  $x_i \in L$  queried by the adversary (note that  $z$  is a valid witness for the statements  $(x_i, y)$ ). The uniqueness property of UWM guarantees that proofs computed by either of the witnesses result in the same proof. Hence, the zero-knowledge property follows. Finally, note that, we can achieve the stronger notion of perfect ZK since the CRS in both the real world and the simulation are identically distributed (both are computed by sampling the CRS  $K$  of UWM and the string  $y$  pseudorandomly).

#### 1.2.5 Perfectly Sound Verifier UWM Implies $\mathbf{NP} = \mathbf{UP}$

Recall that a verifier UWM (VUWM) is similar to an UWM with the additional guarantee that the verifier also accepts a unique proof for each statement  $x$ . The complexity class  $\mathbf{UP}$  consists of problems that are accepted by an unambiguous Turing machine with at most one accepting path for each input. It is easy to see that the verifier of a perfectly sound VUWM acts as a  $\mathbf{UP}$  relation. Also, since we require the VUWM to be perfectly sound it does not require as setup. Hence, it shows that  $\mathbf{NP} \subseteq \mathbf{UP}$ . The other direction is trivial and hence this shows that  $\mathbf{NP} = \mathbf{UP}$ .

## 2 Preliminaries

### 2.1 Notation

For  $n \in \mathbb{N}$ , we write  $[n] = \{1, 2, \dots, n\}$ . If  $x$  is a string, we denote  $|x|$  as the length of  $x$ . For a distribution or random variable  $X$ , we denote  $x \leftarrow X$  the action of sampling an element  $x$  according to  $X$ . When  $A$  is an algorithm, we write  $y \leftarrow A(x)$  to denote a run of  $A$  on input  $x$  and output  $y$ ; if  $A$  is randomized, then  $y$  is a random variable and  $A(x; r)$  denotes a run of  $A$  on input  $x$  and randomness  $r$ . An algorithm  $A$  is probabilistic polynomial-time (PPT) if  $A$  is randomized and for any input  $x, r \in \{0, 1\}^*$ ; the computation of  $A(x; r)$  terminates in at most  $\text{poly}(|x|)$  steps. For a set  $S$ , we let  $U_S$  denote the uniform distribution over  $S$ . For an integer  $\alpha \in \mathbb{N}$ , let  $U_\alpha$  denote the uniform distribution over  $\{0, 1\}^\alpha$ , the bit strings of length  $\alpha$ . Throughout this paper, we denote the security parameter by  $\kappa$ . We refer the reader to the full version of the paper for some basic definitions and concepts related to information theory. We will need the following form of the Chernoff-Hoeffding inequality.

**Lemma 1.** *Let  $S_N$  be the sum of  $N$  independent samples of a Bernoulli random variable, which is 1 with probability  $p$  and 0 otherwise. Then  $\Pr[|S_N - Np| > t] < e^{-t^2/N}$ . In particular,  $\Pr[|S_N - Np| > N^{2/3}] < e^{-N^{1/3}}$ .*

### 2.2 Cryptographic Primitives

Next we summarize some of the cryptographic primitives from literature that we rely on.

#### 2.2.1 Witness PRFs

A witness PRF [17] consists of triple of algorithms  $\text{wPRF} = (\text{wPRF.Gen}, \text{F}, \text{wPRF.Eval})$  as follows:

1.  $\text{wPRF.Gen}(\kappa, R)$  : This is a randomized algorithm that takes as input the security parameter  $1^\kappa$  and the description of a circuit  $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$  and outputs a function secret key  $\text{fk}$  along with a public evaluation key  $\text{ek}$ .
2.  $\text{F}(\text{fk}, x)$  : The private evaluation algorithm  $\text{F}$  is a deterministic algorithm that takes as input the function secret key  $\text{fk}$  and an input  $x \in \mathcal{X}$  and produces some output  $y \in \mathcal{Y}$  for some set  $\mathcal{Y}$ .
3.  $\text{wPRF.Eval}(\text{ek}, x, w)$  : The public evaluation algorithm  $\text{wPRF.Eval}$  is also a deterministic algorithm that takes as input the public evaluation key  $\text{ek}$ , an input  $x \in \mathcal{X}$  and a witness  $w \in \mathcal{W}$  to produce an output  $y \in \mathcal{Y}$  or  $\perp$ .

\* *Correctness.* The correctness of  $\text{wPRF}$  requires that for all  $x \in \mathcal{X}$  and  $w \in \mathcal{W}$ , the following holds:

$$\text{wPRF.Eval}(\text{ek}, x, w) = \begin{cases} \text{F}(\text{fk}, x) & \text{if } R(x, w) = 1 \\ \perp & \text{if } R(x, w) = 0 \end{cases}$$



\* *Security.* We recall the *adaptive instance interactive security* notion for witness PRFs from [17]. Consider the following experiment  $\text{Exp}_{\mathcal{A}}^R(\kappa, b)$  between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ , parameterized by a relation  $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$ , a bit  $b$  and security parameter  $\kappa$ .

- The challenger  $\mathcal{C}$  runs  $(\text{fk}, \text{ek}) \leftarrow \text{wPRF.Gen}(\kappa, R)$ , and gives  $\text{ek}$  to  $\mathcal{A}$ .
- $\mathcal{A}$  can adaptively make queries on instances  $x_i \in \mathcal{X}$  and receives the values  $F(\text{fk}, x_i)$  from  $\mathcal{C}$ .
- At any point in the game,  $\mathcal{A}$  can make a challenge query  $x^* \in \mathcal{X}$ . The challenger computes  $y_0 \leftarrow F(\text{fk}, x^*)$  and  $y_1 \stackrel{\$}{\leftarrow} \mathcal{Y}$ . It then returns  $y_b$  to  $\mathcal{A}$ .
- $\mathcal{A}$  can make additional queries to  $F$  and finally  $\mathcal{A}$  outputs a bit  $b'$ . The challenger  $\mathcal{C}$  checks that  $x^* \notin \{x_i\}$  and that  $x^* \notin L_R$ <sup>3</sup>. If either check fails,  $\mathcal{C}$  outputs a random bit. Otherwise, it outputs  $b'$ .

Let  $W_b$  be the event that the challenger in experiment  $\text{Exp}_{\mathcal{A}}^R(\kappa, b)$  outputs 1. Define the advantage of  $\mathcal{A}$  as  $\text{wPRF.Adv}_{\mathcal{A}}^R(\kappa) = |\Pr[W_0] - \Pr[W_1]|$ .

**Definition 1 (Adaptive instance Interactive security).**  $\text{wPRF} = (\text{wPRF.Gen}, F, \text{wPRF.Eval})$  is *adaptive instance interactively secure* for a NP relation  $R$  if for all PPT adversaries  $\mathcal{A}$ , the advantage  $\text{wPRF.Adv}_{\mathcal{A}}^R$  of  $\mathcal{A}$  is negligible in the security parameter  $\kappa$ .

One can also define *non-interactive* security for witness PRFs, where the adversary  $\mathcal{A}$  in the above experiment is not allowed to make any  $F$  queries.

**Definition 2 (Adaptive instance Non-Interactive security).**  $\text{wPRF} = (\text{wPRF.Gen}, F, \text{wPRF.Eval})$  is *adaptive instance non-interactively secure* for a NP relation  $R$  if for all PPT adversaries  $\mathcal{A}$ , the advantage  $\text{wPRF.Adv}_{\mathcal{A}}^R$  of  $\mathcal{A}$  is negligible in the security parameter  $\kappa$ , and additionally the adversary  $\mathcal{A}$  is not allowed to make any  $F$  queries in the above experiment.

Finally, one can also define a weaker notion of security, called the *static instance interactive (non-interactive) security*, where the adversary needs to commit to the challenge  $x^*$  before seeing  $\text{ek}$  or before making any queries to the oracle  $F$ . We note that one can convert any static-instance interactive (resp. non-interactive) witness PRF to an adaptive instance one by relying on complexity leveraging when appropriate.

### 2.2.2 Generalized Goldreich-Levin Theorem

For our construction of witness PRF from witness maps we will need to use a generalized version of the Goldreich-Levin (GL) theorem [7], as stated below.

**Lemma 2 (Generalized Goldreich-Levin Theorem).** *There exists a PPT inverter  $\mathcal{A}'$  and a non-zero polynomial  $q(\cdot)$  such that, for any PPT algorithm  $\mathcal{A}$  and any  $(\alpha, \beta) \in \{0, 1\}^k \times \{0, 1\}^\ell$  such that  $p(\alpha) := \Pr[\mathcal{A}(\alpha, r) = \langle \beta, r \rangle : r \stackrel{\$}{\leftarrow} \{0, 1\}^\ell]$  (where  $\langle \cdot, \cdot \rangle$  denotes the inner product over the binary field), then  $\Pr[\mathcal{A}'^{\mathcal{A}(\alpha, \cdot)}(1^\ell, \alpha) = \beta] \geq q(p(\alpha) - \frac{1}{2})$ .*

<sup>3</sup> Note that, the language  $L_R := \{x \mid \exists w, (x, w) \in R\}$ .

### 2.2.3 Leakage-resilient Weak PRF

A standard weak PRF (wPRF) requires that given arbitrarily many uniformly random inputs  $x_1, \dots, x_q$ , the outputs of the wPRF  $y_1, \dots, y_q$  look pseudorandom. A leakage-resilient wPRF (LR-wPRF) requires wPRF security to hold even if the attacker can leak some information about the secret key. In particular, we will consider the entropy-bounded leakage model [3, 5]. Following [5], we first recall the notion of  $\lambda$ -leaky functions.

**Definition 3 ( $\lambda$ -leaky function).** *A probabilistic function  $h : \{0, 1\}^* \rightarrow \{0, 1\}^*$  is  $\lambda$ -leaky if, for all  $n \in \mathbb{N}$  we have  $\tilde{H}_\infty(\mathcal{U}_n | h(\mathcal{U}_n)) \geq n - \lambda$ , where  $\mathcal{U}_n$  is the uniform distribution over  $\{0, 1\}^n$ .*

As shown in [5], if a function is  $\lambda$ -leaky (decreases the entropy of the uniform distribution by at most  $\lambda$  bits), then it decreases the entropy of every distribution by at most  $\lambda$  bits. Moreover, the definition composes nicely and an adversary that adaptively chooses several  $\lambda_i$ -leaky functions, only learns  $\sum_i \lambda_i$  bits of information.

Informally, we say a function  $\mathcal{F}_K(\$)$  is a  $\lambda(\kappa)$ -leakage-resilient weak PRF in the entropy-bounded leakage model, if the weak PRF security guarantee is maintained, even if the adversary can learn the output of a  $\lambda$ -leaky function on the key  $K$ . We refer to the full version for the formal definition.

We also rely on statistically-sound NIWI proof systems. We refer to the full version for the formal definition.

## 3 Different Notions of Witness Maps and Their Definitions

In this section, we present the definition of compact witness map from [4]. We then present different variations of their definition, namely designated-verifier witness maps and verifier-compact witness maps, which we introduce in this work. We start by recalling the definition of compact witness map (CWM) from [4].

### 3.1 Compact Witness Maps

We say that  $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$  is said to be an **NP** relation if membership in it can be computed in time polynomial in the length of the first input. Given an **NP** relation  $R$ , we define the **NP** language  $L_R := \{x \mid \exists w, (x, w) \in R\}$ . When referring to  $(x, w) \in R$ , where  $R$  is a given **NP** relation,  $x$  is called the statement and  $w$  the witness. It will be convenient for us to consider **NP** relations parametrized with their input length: Below we let  $R_\ell := R \cap \{0, 1\}^\ell \times \{0, 1\}^*$ .

**Definition 4 (Compact Witness Map (CWM)).** For  $\alpha \geq 0$ , an  $\alpha$ -CWM for an NP relation  $R$  is a triple  $\text{CWM} = (\text{setup}, \text{map}, \text{check})$  where  $\text{setup}$  is a PPT algorithm and the other two are deterministic polynomial time algorithms such that:

- $\text{setup}(\kappa, \ell)$  outputs a string  $K$  of length polynomial in the security parameter  $\kappa$  and  $\ell$ , where  $\ell = \ell(\kappa)$  is an upper bound on the length of the statements supported by CWM.
- Completeness: For any polynomial  $\ell$ ,  $\forall (x, w) \in R_{\ell(\kappa)}$ ,  $\forall K \leftarrow \text{setup}(\kappa, \ell(\kappa))$ ,

$$\text{check}(K, x, \text{map}(K, x, w)) = 1.$$

- Compactness: For any polynomial  $\ell$ ,  $\forall K \leftarrow \text{setup}(\kappa, \ell(\kappa))$ ,  $\forall x \in \{0, 1\}^{\ell(\kappa)}$ ,

$$|\{\text{map}(K, x, w) \mid (x, w) \in R_{\ell(\kappa)}\}| \leq 2^\alpha.$$

- Soundness: For any polynomial  $\ell$  and any PPT adversary  $\mathcal{A}$ ,  $\text{Adv}_{\mathcal{A}}^{\text{CWM}}(\kappa)$  defined below is negligible:

$$\Pr_{\substack{K \leftarrow \text{setup}(\kappa, \ell(\kappa)) \\ (x^*, w^*) \leftarrow \mathcal{A}(K)}} [\text{check}(K, x^*, w^*) = 1, x^* \notin L_R].$$

A 0-CWM is also called a **Unique Witness Map (UWM)**.

The above definition has perfect security in the sense that the completeness and compactness conditions hold for every possible  $K$  that  $\text{CWM.setup}$  can output with positive probability. A statistical version, where this needs to hold with all but negligible probability over the choice of  $K$  will suffice for all our applications. But for simplicity, we shall use the perfect version above.

### 3.2 Designated-Verifier Witness Maps

In this section, we define (reusable/non-reusable) designated-verifier compact witness maps (dv-CWM).

**Definition 5 (Reusable/Non-Reusable dv-CWM).** For  $\alpha \geq 0$ , a reusable (resp. non-reusable)  $\alpha$ -dv-CWM for an NP relation  $R$  is a triple  $\text{DV-CWM} = (\text{setup}, \text{map}, \text{check})$  where  $\text{setup}$  is a PPT algorithm that on input the security parameter  $\kappa$  and the statement length  $\ell$ , outputs a pair of strings  $(K, \text{VK})$ , and the other two are deterministic polynomial time algorithms that satisfy the completeness, compactness and reusable soundness (resp. non-reusable soundness) conditions below.

- Completeness: For any polynomial  $\ell$ ,  $\forall (x, w) \in R_{\ell(\kappa)}$ ,  $\forall (K, \text{VK}) \leftarrow \text{setup}(\kappa, \ell(\kappa))$ ,

$$\text{check}(\text{VK}, x, \text{map}(K, x, w)) = 1.$$

- Compactness: For any polynomial  $\ell$ ,  $\forall(\mathbf{K}, \mathbf{VK}) \leftarrow \text{setup}(\kappa, \ell(\kappa))$ ,  $\forall x \in \{0, 1\}^{\ell(\kappa)}$ ,
 
$$|\{\text{map}(\mathbf{K}, x, w) \mid (x, w) \in R_{\ell(\kappa)}\}| \leq 2^\alpha.$$
- Reusable/Non-Reusable Soundness: Reusable soundness requires that the advantage  $\text{Adv}_{\mathcal{A}}^{\text{DV-CWM}}(\kappa)$  defined below is negligible for every polynomial  $\ell$  and every PPT adversary  $\mathcal{A}$  with oracle access to  $\text{check}(\mathbf{VK}, \cdot, \cdot)$ .

$$\text{Adv}_{\mathcal{A}}^{\text{DV-CWM}}(\kappa) = \Pr_{\substack{(\mathbf{K}, \mathbf{VK}) \leftarrow \text{setup}(\kappa, \ell(\kappa)) \\ (x^*, w^*) \leftarrow \mathcal{A}^{\text{check}(\mathbf{VK}, \cdot, \cdot)}(\mathbf{K})}} [\text{check}(\mathbf{VK}, x^*, w^*) = 1 \wedge x^* \notin L_R].$$

Non-reusable soundness requires that  $\text{Adv}_{\mathcal{A}}^{\text{DV-CWM}}(\kappa)$  is negligible for every polynomial  $\ell$  and every PPT adversary  $\mathcal{A}$  which does not access its oracle.

A 0-dv-CWM is also called a Designated-verifier UWM (dv-UWM).

Similar to CWM, one can also define a weaker notion of selective reusable (resp. non-reusable) soundness, in which the adversary is required to generate  $x^*$  first (given  $\kappa, \ell$ ) before it gets  $\mathbf{K}$  and access to the verification oracle  $\text{check}(\mathbf{VK}, \cdot, \cdot)$ .

### 3.3 Verifier-Compact Witness Maps

**Definition 6 (Verifier-Compact Witness Maps (VCWM)).** For  $\alpha \geq 0$ , an  $\alpha$ -VCWM  $(\text{setup}, \text{map}, \text{check})$  for an NP relation  $R$  is a CWM for  $R$  satisfying the following additional condition:

- Verifier-Compactness: For any polynomial  $\ell$ ,  $\forall \mathbf{K} \leftarrow \text{setup}(\kappa, \ell(\kappa))$ ,  $\forall x \in \{0, 1\}^{\ell(\kappa)}$ ,
 
$$|\{w^* \mid \text{check}(\mathbf{K}, x, w^*) = 1\}| \leq 2^\alpha.$$

A 0-VCWM is also called a verifier-unique witness map (vUWM).

**Selective Soundness.** The soundness condition for CWM and its variants (dv-CWM and VCWM) can be relaxed to obtain a *selectively sound* variant of the corresponding primitive. In the soundness conditions above, we considered an adversary  $\mathcal{A}$  which outputs a statement  $x^*$  and a purported proof  $w^*$  at the end of the experiment. For selective soundness, we require  $\mathcal{A}$  to output  $x^*$  at the beginning (given only  $\kappa, \ell$ ), before  $\text{setup}$  is executed. This level of soundness suffices for some applications (e.g., construction of a witness encryption scheme from a UWM), as shown in [4]. It also provides an intermediate target for constructions, as one can convert a selectively sound CWM to a standard CWM by relying on complexity leveraging.

## 4 Equivalence of Witness PRFs and dv-UWM

In this section, we explore the relationship between witness PRF (wPRF) and unique witness maps. In particular, we show that witness PRF and designated-verifier UWM (dv-UWM) are equivalent for **NP**. For these implications, we consider the static security variant of witness PRF and selective soundness of dv-UWM. Our implications can be adapted to the adaptive security variants of both these notions via complexity leveraging.

### 4.1 Witness PRF Imply dv-UWM

In this section, we present the construction of our (selective reusable sound) dv-UWM for any **NP** relation  $R$ . The main building block of our construction is a (static-instance secure) witness PRF for  $R$ .

**Construction.** Let  $\text{wPRF} = (\text{wPRF.Gen}, \text{F}, \text{wPRF.Eval})$  be a *static-instance interactively secure* witness PRF for any **NP** relation  $R$  parametrized by statements of length at most  $\ell(\kappa)$ , where  $\kappa$  is the security parameter and  $\ell$  is an arbitrary (but fixed) polynomial in the security parameter. We construct a (selective) *reusable* dv-UWM  $\text{DV-UWM} = (\text{dv.setup}, \text{dv.map}, \text{dv.check})$  for  $R$  as follows:

- $\text{dv.setup}(\kappa, \ell) : \text{Run } (\text{fk}, \text{ek}) \leftarrow \text{wPRF.Gen}(\kappa, R)$ . Set  $\text{K} = \text{ek}$  and  $\text{VK} = \text{fk}$ .
- $\text{dv.map}(\text{K}, x, w) : \text{Parse } \text{K} \text{ as } \text{ek}$ . Run  $y = \text{wPRF.Eval}(\text{ek}, x, w)$ . Output  $w^* = y$
- $\text{dv.check}(\text{VK}, x, w^*) : \text{Parse } \text{VK} \text{ as } \text{fk}$  and compute  $y' = \text{F}(\text{fk}, x)$  and check if  $w^* \stackrel{?}{=} y'$ . If the check is satisfied, output 1; else output  $\perp$ .

**Theorem 1.** *Let wPRF be an static-instance interactively secure witness PRF for **NP** with super-polynomial range  $|\mathcal{Y}| = \kappa^{\omega(1)}$ . Then the above construction of dv-UWM for **NP** satisfies selective reusable soundness.*

*Proof.* Firstly, we note that DV-UWM satisfies perfect completeness (assuming wPRF is perfectly correct). Also, it satisfies uniqueness, since  $(x, w)$  is deterministically mapped to the output of wPRF, regardless of the witness  $w$ . In particular, the correctness of wPRF guarantees that for all  $(x, w) \in R$ ,  $\text{wPRF.Eval}(\text{ek}, x, w) = \text{F}(\text{fk}, x)$ . The later function (i.e.,  $\text{F}(\text{fk}, \cdot)$ ) does not depend on  $w$  and deterministically maps  $x$  to a unique output value  $y \in \mathcal{Y}$ . Below, we shall prove that the construction satisfies selective reusable soundness as well.

Consider an adversary  $\mathcal{A}$  in the definition of  $\text{Adv}_{\mathcal{A}}^{\text{DV-UWM}}(\kappa)$  (see Def. 5). Note that, in the (selective) reusable soundness experiment the adversary  $\mathcal{A}$  first commits to the challenge  $x^*$  and then gets access to the public parameter  $\text{K}$  and the verification oracle, namely  $\text{dv.check}(\text{VK}, \cdot, \cdot)$ . The oracle takes as input tuples of the form  $(x_i, w_i^*)$  and outputs either 1 or 0. We show how to construct another adversary  $\mathcal{B}$  breaking the static-instance interactive security of wPRF using  $\mathcal{A}$  in a black-box way. The adversary  $\mathcal{B}$  simulates the environment of  $\mathcal{A}$  as follows:

1. The adversary  $\mathcal{B}$  first commits to a challenge  $x^*$  such that  $x^* \notin L_R$ . The adversary  $\mathcal{A}$  forwards  $x^*$  to its own challenger and receives a value  $y^* \in \mathcal{Y}$ , which it stores in its memory.

2. The adversary  $\mathcal{B}$  then receives  $\text{ek}$  from its challenger and sets  $\text{K} = \text{ek}$ . It gives  $\text{K}$  to  $\mathcal{A}$ .
3. When  $\mathcal{A}$  queries a tuple  $(x_i, w_i^*)$  to  $\text{dv.check}(\text{K}, \text{VK}, \cdot, \cdot)$ , the adversary  $\mathcal{B}$  does the following:
  - Query the oracle  $\text{F}(\text{fk}, \cdot)$  on input  $x_i$  to receive some output  $y_i$ .
  - Checks if  $w_i^* \stackrel{?}{=} y_i$ . If so, it outputs 1 to  $\mathcal{A}$  else it outputs 0.
4. Finally, at some point  $\mathcal{A}$  outputs  $w^*$  corresponding to the challenge  $x^*$  (which it committed to before). The adversary  $\mathcal{A}$  then retrieves the value  $y^*$  from its memory and checks if  $y^* \stackrel{?}{=} w^*$ . If the check passes,  $\mathcal{B}$  outputs 0; else it outputs 1.

This completes the description of simulation of  $\mathcal{A}$ 's environment by  $\mathcal{B}$ . Let us assume that  $\mathcal{A}$  makes a total of  $q = q(\kappa)$  queries to the verification oracle  $\text{dv.check}(\text{VK}, \cdot, \cdot)$  (including the challenge query). Since  $\mathcal{A}$  has some non-negligible advantage (say  $\epsilon$ ) in breaking the selective reusable soundness of DV-UWM, it must hold that  $\text{dv.check}(\text{VK}, x^*, w^*) = 1$  holds with probability  $\epsilon$ . According to the construction, the above check passes whenever  $w^* = \text{F}(\text{fk}, x^*)$ . If the value  $y^*$  received by  $\mathcal{B}$  from its challenger was computed using the function  $\text{F}(\text{fk}, \cdot)$ , then it always holds that  $y^* = w^*$ . However, if  $y^*$  was randomly sampled, the probability that  $w^*$  is equal to  $y^*$  is  $\frac{1}{|\mathcal{Y}|}$ , which is negligible. Hence, the advantage of  $\mathcal{B}$  in breaking the adaptive-instance interactive security of wPRF is  $\epsilon - \frac{\epsilon}{|\mathcal{Y}|}$ , which is non-negligible, thereby contradicting the security of wPRF.  $\square$

## 4.2 dv-UWM Implies Witness PRF

Now, we present our implication in the other direction, namely that dv-UWM for **NP** implies witness PRF for **NP**. We split our transformation into two phases. First, we present a construction of a (static) non-interactive witness PRF for **NP** from any (selective) non-reusably sound dv-UWM for **NP**. Next, we show a generic transformation from any (static) non-interactive witness PRF for **NP** to an (static) interactive witness PRF for **NP** additionally using any one way function.

### 4.2.1 dv-UWM Implies Non-interactive Witness PRF

In this section, we show our first transformation from any (selective) non-reusably sound dv-UWM for **NP** to a (static-instance) non-interactive witness PRF for **NP**. The construction is shown in Fig. 2.

**Theorem 2.** *If DV-UWM is a selective non-reusably sound dv-UWM for the **NP** relation  $\mathfrak{R}$  (defined in Fig. 2), then the construction shown in Fig. 2 is a static-instance non-interactively secure witness PRF for the **NP** relation  $R$ .*

Let  $R$  be a **NP** relation, and  $L_R$  be the corresponding **NP** language defined as  $L_R := \{x \mid \exists w : (x, w) \in R\}$ . Let  $R'$  be another **NP** relation defined as  $(y, z) \in R'$  if and only if  $y = G(z)$ , where  $G : \{0, 1\}^\kappa \rightarrow \{0, 1\}^{2\kappa}$  is a length-doubling pseudo-random generator. Also, let  $L_{R'}$  be the corresponding **NP** language defined as  $L_{R'} := \{y \mid \exists z : (y, z) \in R'\}$ . Further, we assume that  $R$  and  $R'$  are parameterized with their input lengths. Define the following derived **NP** relation  $\mathfrak{R}$  and language  $L_{\mathfrak{R}}$  as:

$$\mathfrak{R}((x, y), (w, z)) = 1 \iff R(x, w) = 1 \vee R'(y, z) = 1, \text{ and}$$

$$L_{\mathfrak{R}} = \{(x, y) \mid \exists (w, z), ((x, y), (w, z)) \in \mathfrak{R}\}.$$

Note that, the relation  $\mathfrak{R}$  is parameterized with statements of length at most  $\ell' = \ell + 2\kappa$ .

- (a) Let  $\text{DV-UWM} = (\text{dv.setup}, \text{dv.map}, \text{dv.check})$  be a (selectively) sound  $\text{dv-UWM}$  for the language  $L_{\mathfrak{R}}$ . Further, let the length of the representative  $w^*$  of  $\text{DV-UWM}$  be  $p(\kappa)$  bits, for some polynomial  $p(\cdot)$ .
  - (b) Let  $\text{GL}(\pi, r)$  denote the Goldreich-Levin (GL) hardcore bit [7] of  $\pi$  using randomness  $r$ . Recall that, the GL predicate is the bit-wise inner product of  $\pi$  and  $r$ .
1.  $\text{wPRF.Gen}(\kappa, R)$  : Takes as input an **NP** relation  $R$  (parametrized by its input length  $\ell$ ) as defined above. Run  $(\mathsf{K}, \mathsf{VK}) \leftarrow \text{dv.setup}(\kappa, \ell')$ , where  $\ell'$  is defined as above. Sample  $z \leftarrow \{0, 1\}^\kappa$  and  $r \leftarrow \{0, 1\}^{p(\kappa)}$  uniformly at random, and compute  $y = G(z)$ . Set  $\text{ek} = (\mathsf{K}, y, r)$  and  $\text{fk} = z$ .
  2.  $\text{F}(\text{fk}, x)$  : Takes as input an instance  $x \in L_R$ . It does the following:
    - Computes the representative witness  $w^* = \text{dv.map}(\mathsf{K}, (x, y), (\perp, z))$ , using  $(\perp, z)$  as witness. Note that,  $(x, y) \in L_{\mathfrak{R}}$ .
    - Compute the GL hardcore bit  $b = \text{GL}(w^*, r)$ .
  3.  $\text{wPRF.Eval}(\text{ek}, x, w)$  : Takes as input an instance  $x \in L_R$ . It does the following:
    - Computes a representative witness  $w^* = \text{dv.map}(\mathsf{K}, (x, y), (w, \perp))$ , using  $(w, \perp)$  as witness. Note that,  $(x, y) \in L_{\mathfrak{R}}$ .
    - Compute the GL hardcore bit  $b = \text{GL}(w^*, r)$ .

**Fig. 2.** Construction of Non-Interactive Witness PRF  $\text{wPRF}$  from  $\text{DV-UWM}$

*Proof.* We show that any adversary  $\mathcal{A}_{\text{wprf}}$  breaking the static-instance non-interactive security of  $\text{wPRF}$  with a noticeable advantage can be transformed into an adversary  $\mathcal{A}_{\text{dv-uwmm}}$  breaking the selective non-reusable soundness of  $\text{DV-UWM}$ . Note that, in the static-instance non-interactive security game of  $\text{wPRF}$  the adversary commits to the challenge  $x^*$  before seeing the evaluation key  $\text{ek}$ . At first, we show that the adversary  $\mathcal{A}_{\text{wprf}}$  breaking the security of  $\text{wPRF}$  can be converted into a predictor  $\mathcal{A}_{\text{GL}}$  for the generalized Goldreich-Levin theorem.

In more details, let the adversary  $\mathcal{A}_{\text{wprf}}$  can predict the bit  $b$  in the security experiment of wPRF on the challenge instance  $x^* \notin L_R$  with non-negligible probability. Let,  $\alpha = (\mathbf{K}, y)$ , and  $\beta = w^*$  from the generalized GL theorem (see Lemma 2). This implies that we can construct a distinguisher  $\mathcal{A}$  that on input  $(\alpha = (\mathbf{K}, y), r)$  can distinguish the bit  $b$  (in the above construction) from a random bit with non-negligible probability. Hence, by Lemma 2, we can use this distinguisher  $\mathcal{A}$  to construct a predictor  $\mathcal{A}'$ , who given  $\alpha = (\mathbf{K}, y)$  can predict the pre-image  $w^*$  with non-negligible probability. This implies that the predictor outputs  $w^*$  such that  $w^* = \text{dv.map}(\mathbf{K}, (x, y), (\perp, z))$  with non-negligible probability. At this point, instead of computing  $y = G(z)$ , we sample a random  $y \leftarrow \{0, 1\}^{2\kappa}$ . The security of the PRG  $G$  ensures that this switch is indistinguishable to  $\mathcal{A}'$ . Hence the probability that  $\mathcal{A}'$  outputs a “valid”  $w^*$  ( $w^*$  such that  $\text{dv.check}(\mathbf{K}, \mathbf{VK}, (x, y), w^*) = 1$ ) continues to hold, except with a negligible probability. However, note that, with very high probability it holds that  $(x, y) \notin \mathfrak{R}$ . This contradicts the selective non-reusable soundness property of dv-UWM, since the adversary  $\mathcal{A}'$  outputs a valid representative witness corresponding to a false statement  $(x, y) \notin \mathfrak{R}$ .  $\square$

### 4.2.2 Equivalence of Non-Interactive and Interactive Witness PRF

In this section, we show that any (static-instance) non-interactive witness PRF for NP can be generically transformed to a (static-instance) interactive witness PRF for NP. The construction is given in Fig. 3.

**Theorem 3.** *Let nl-wPRF be a static-instance non-interactively secure witness PRF for the NP relation  $\Psi$  (defined in Fig. 3), and (Com, Open) be a statistically binding commitment scheme. Then the construction shown in Fig. 3 is a static-instance interactively secure witness PRF for the NP relation  $\mathcal{R}$ .*

The detailed proof of this theorem is presented in the full version of the paper. The intuition behind the proof is given in the technical overview.

## 5 Extremely Compact WM Implies Pseudo-UWM

In this section, we show that an *extremely compact* WM – i.e., a CWM with polynomial-sized image – implies a Pseudo-UWM (p-UWM) where the uniqueness may not hold with an inverse polynomial probability (that can be made arbitrarily small). In other words, we show that a  $\alpha$ -CWM for  $\alpha = O(\log \kappa)$  for security parameter  $\kappa$  implies a p-UWM as defined below.

**Definition 7 (Pseudo Unique Witness Map (p-UWM)).** *A Pseudo-UWM (p-UWM) for an NP relation  $R$  is a triple  $\text{CWM} = (\text{setup}, \text{map}, \text{check})$  where setup is a PPT algorithm, and map and check are deterministic polynomial time algorithms such that:*



Let  $\mathcal{R} : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$  be an **NP** relation restricted to inputs in  $\mathcal{X} = \{0, 1\}^{\ell(\kappa)}$ , and  $L_{\mathcal{R}}$  be the corresponding **NP** language defined as  $L_{\mathcal{R}} := \{x \mid \exists w, (x, w) \in \mathcal{R}\}$ . Let  $\mathcal{R}'$  be another **NP** relation defined as:

$$((x, c), (x', r)) \in \mathcal{R}' \iff (c = \text{Com}(x'; r) \wedge (x \neq x')),$$

where **Com** is a polynomial-time statistically binding commitment scheme over a message space  $\mathcal{X} \cup \{\mathbf{0}\}$ , where  $\mathbf{0} \notin \mathcal{X}$ . Also, let  $L_{\mathcal{R}'}$  be the corresponding **NP** language. Finally, let us define the following **NP** relation  $\Psi$  and language  $L_{\Psi}$  as:

$$((x, c), (w, x', r)) \in \Psi \iff (x, w) \in \mathcal{R} \vee ((x, c), (x', r)) \in \mathcal{R}' \text{ and } L_{\Psi} = \{(x, c) \mid \exists (w, x', r) \text{ s.t. } ((x, c), (w, x', r)) \in \Psi\}.$$

Note that, the relation  $\Psi$  is parameterized with the input length  $\ell' = \ell + |c|$ .

- Let **nl-wPRF** = (**nl-wPRF.Gen**, **nl-wPRF.F**, **nl-wPRF.Eval**) be a static-instance non-interactive witness PRF for the NP relation  $\Psi$  defined above.

We construct a static-instance interactively secure witness PRF **wPRF** = (**wPRF.Gen**, **F**, **wPRF.Eval**) as follows:

1. **wPRF.Gen**( $\kappa, \mathcal{R}$ ) : Takes as input the **NP** relation  $\mathcal{R}$  defined above. Run  $(\hat{ek}, \hat{fk}) \leftarrow \text{nl-wPRF.Gen}(\kappa, \Psi)$ , where the relation  $\Psi$  is defined as above. Sample a random tape  $r$  for the commitment scheme **Com**, and compute  $c = \text{Com}(\mathbf{0}; r)$ . Set  $ek = (\hat{ek}, c)$  and  $fk = (\hat{fk}, r)$ .
2. **F**( $fk, x$ ) : Parse  $fk$  as  $(\hat{fk}, r)$ , compute the commitment  $c = \text{Com}(\mathbf{0}; r)$  and output  $y = \text{nl-wPRF.F}(\hat{fk}, (x, c))$ .
3. **wPRF.Eval**( $ek, x, w$ ) : Takes as input an instance-witness pair  $(x, w) \in \mathcal{R}$  and does the following: Parse  $ek$  as  $(\hat{ek}, c)$  and output  $\text{nl-wPRF.Eval}(\hat{ek}, (x, c), (w, \perp, \perp))$ . Note that  $(x, c) \in L_{\Psi}$ .

**Fig. 3.** Construction of an interactive wPRF wPRF from a non-interactive wPRF nl-wPRF

- **setup**( $\kappa, \ell, \epsilon$ ) outputs a string  $K$  of length polynomial in  $\kappa, \ell$  and  $1/\epsilon$ .
- Completeness: For any polynomials  $\ell, 1/\epsilon, \forall (x, w) \in R_{\ell(\kappa)}$ ,

$$\Pr_{\substack{K \leftarrow \text{setup}(\kappa, \ell(\kappa), \epsilon(\kappa)) \\ w^* \leftarrow \text{map}(K, x, w)}} [\text{check}(K, x, w^*) = 1] = 1.$$

- Pseudo-Uniqueness: For any polynomial  $\ell, \forall x \in \{0, 1\}^{\ell(\kappa)}, \forall w_1, w_2$  such that  $(x, w_1), (x, w_2) \in R_{\ell(\kappa)}$  (possibly  $w_1 = w_2$ ),

$$\Pr_{\substack{K \leftarrow \text{setup}(\kappa, \ell(\kappa), \epsilon(\kappa)) \\ w_1^* \leftarrow \text{map}(K, x, w_1) \\ w_2^* \leftarrow \text{map}(K, x, w_2)}} [w_1^* \neq w_2^*] \leq \epsilon(\kappa)$$

- *Soundness*: For any polynomials  $\ell, 1/\epsilon$ , and any PPT adversary  $\mathcal{A}$ ,  $\text{Adv}_{\mathcal{A}}^{\text{CWM}}(\kappa)$  defined below is negligible:

$$\Pr_{\substack{\kappa \leftarrow \text{setup}(\kappa, \ell(\kappa), \epsilon(\kappa)) \\ (x^*, w^*) \leftarrow \mathcal{A}(\kappa)}}} [\text{check}(\mathsf{K}, x^*, w^*) = 1, x^* \notin L_R].$$

We now present the construction below. The main building block of our construction is a statistically-sound NIWI (SNIWI) argument system NIWI. Before proceeding with the construction, let us try to solve a seemingly unrelated algorithmic problem, which we call the *pseudo-deterministic sampling* (PDS) problem over a polynomial-sized domain. We will later see how to use a solution for the PDS problem in our construction of p-UWM from CWM.

**Pseudo-Deterministic Sampling for Small Domains.** Let  $\mathcal{D}$  be an arbitrary distribution over some set  $X$  with a support of size  $n$ . Our goal is to design an algorithm  $\text{PDS.sam}$  that is polynomial-time in  $n$  and with only sampling access to  $\mathcal{D}$ , can *pseudo-deterministically* output an element from the support of  $\mathcal{D}$ .  $\text{PDS.sam}$  takes a reference string  $\text{crs}$ , and the pseudo-determinism is required to hold with high probability over the choice of  $\text{crs}$ . More formally,  $(\text{PDS.setup}, \text{PDS.sam})$  is said to be a PDS scheme if:

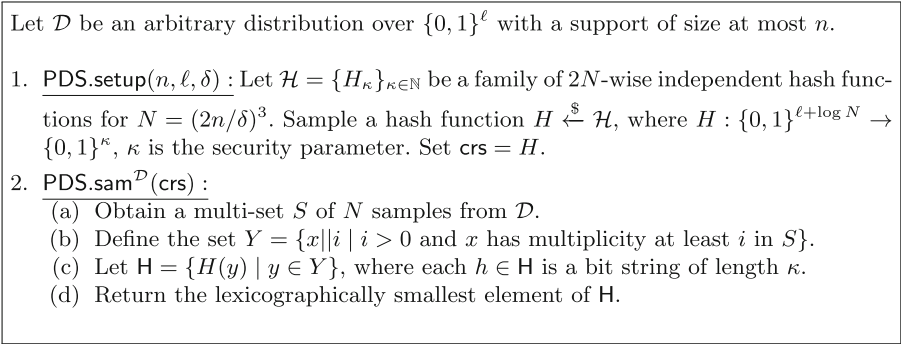
1.  $\text{PDS.setup}(n, \ell, \delta)$ , with inputs the security parameter  $\kappa$ , a bound  $n$  on the support size of distributions over  $\{0, 1\}^\ell$  that are to be handled, and a probability  $\delta > 0$ , outputs a common reference string  $\text{crs}$  of length polynomial in  $n, \ell$  and  $1/\delta$ .
2.  $\text{PDS.sam}^{\mathcal{D}}(\text{crs})$ : Given an input  $\text{crs}$  and sampling access to a distribution  $\mathcal{D}$  over  $\{0, 1\}^\ell$ , the algorithm  $\text{PDS.sam}$  gets a polynomial number of samples from  $\mathcal{D}$  (polynomial in  $|\text{crs}|$ ) and outputs an element in the support of  $\mathcal{D}$ , such that the following holds:
  - **Pseudo-determinism**: For all distributions  $\mathcal{D}$  over  $\{0, 1\}^\ell$  with support size at most  $n$ ,

$$\Pr_{\substack{\text{crs} \leftarrow \text{PDS.setup}(n, \ell, \delta) \\ c_1 \leftarrow \text{PDS.sam}^{\mathcal{D}}(\text{crs}), c_2 \leftarrow \text{PDS.sam}^{\mathcal{D}}(\text{crs)}}} [c_1 \neq c_2] \leq \delta$$

where the probability is over the choice of  $\text{crs}$  as well as the samples from  $\mathcal{D}$ .

We refer to an  $(n, \ell, \delta)$ -PDS as a PDS scheme setup with those parameters.

We now present a construction of a PDS scheme in Fig. 4.



**Fig. 4.** A pseudo-deterministic sampling algorithm for a small support

**Lemma 3.** (PDS.setup, PDS.sam) (Fig. 4) is a PDS scheme.

*Proof.* Since PDS.sam satisfies the efficiency requirements, and always outputs an element in the support of  $\mathcal{D}$ , it remains to show that it satisfies the pseudo-determinism requirement.

Consider two independent runs of PDS.sam using the same  $H$ . Let  $S_i, Y_i$  denote the multi-set of samples and the set of count-appended samples in the two executions.

First, fix the sets  $Y_1$  and  $Y_2$ . Note that if the lexicographically smallest element in  $\{H(y) \mid y \in Y_1 \cup Y_2\}$  is an element  $H(y)$  for  $y \in Y_1 \cap Y_2$ , then PDS.sam outputs the same value in both runs. Now, since  $\mathcal{H}$  is  $2N$ -wise independent,  $H$  behaves identical to a random function over  $Y_1 \cup Y_2$ . Hence, over the choice of  $H$ , the probability of error – i.e., that the outputs of PDS.sam are different – is upper bounded by  $\frac{|Y_1 \Delta Y_2|}{|Y_1 \cup Y_2|} \leq \frac{|Y_1 \Delta Y_2|}{N}$ .

Now we define the following “Good” event for the choice of  $(S_1, S_2)$  over the samples from  $\mathcal{D}$  (independent of  $H$ ): For all  $x$  the support of  $\mathcal{D}$ , the multiplicity of  $x$  in  $S_1$  and in  $S_2$  are both in the range  $[Np - N^{2/3}, Np + N^{2/3}]$ , where  $p$  is the probability assigned to  $x$  by  $\mathcal{D}$ . When this condition holds,  $|Y_1 \Delta Y_2| \leq n \cdot 2N^{2/3}$ , where  $n$  is an upper bound on the size of the support of  $\mathcal{D}$ . Hence, conditioned on the Good event, the probability of error is at most  $\frac{|n \cdot 2N^{2/3}|}{2N}$ .

Finally, by Lemma 1 and union bound (for each  $x$  in the support of  $\mathcal{D}$ , and each of  $Y_1, Y_2$ ), the probability of the Good event not occurring is at most  $2ne^{-N^{1/3}}$ . Hence the probability of error is at most  $2ne^{-N^{1/3}} + nN^{-1/3}$ . Letting  $N = (2n/\delta)^3$ , this is at most  $\delta$ . □

Now we present the p-UWM scheme for an NP relation  $R$ . For simplicity, first we present a randomized map, and then point out how to derandomize it using the CRS.

1. The p-UWM setup outputs  $\text{crs} = (\text{crs}_{\text{NIWI}}, \text{crs}_{\text{CWM}}, \text{crs}_{\text{PDS.sam}})$ , which consists of the setup for a statistically sound NIWI proof system for the relation  $R$ , an  $\alpha$ -CWM for the relation corresponding to the NIWI verifier, and a  $(2^\alpha, \ell, \epsilon/2)$ -PDS scheme (as given above).
2. The p-UWM, on input  $(x, w) \in R$ , defines the distribution  $\mathcal{D}_{x,w}$  as the distribution of  $\text{map}(\text{NIWI}(x, w; \text{crs}_{\text{NIWI}}); \text{crs}_{\text{CWM}})$ . Then it outputs  $\text{PDS.sam}^{\mathcal{D}_{x,w}}$ .
3. The p-UWM verifier is the same as `check`.

Completeness is easy to see. The soundness of this proof depends on the fact that for any false statement, there does not *exist* a NIWI proof that gets accepted (except with negligible probability, over the choice of  $\text{crs}_{\text{NIWI}}$ ), and that CWM is (computationally) sound. For pseudo-uniqueness, first consider two runs of p-UWM (with the same setup) using the same witness  $w$ . In this case, from the compactness of CWM and the pseudo-determinism of the PDS scheme, the probability of the outputs differing is at most  $\epsilon/2$ . In the general case, when two different witnesses  $w_1, w_2$  are used, we note that the distributions  $\mathcal{D}_{x,w_1}$  and  $\mathcal{D}_{x,w_2}$  are computationally indistinguishable from each other, thanks to the witness indistinguishability property of NIWI. Since `PDS.sam` is computationally efficient, this implies that the probability of the outputs differing given access to  $\mathcal{D}_{x,w_1}$  and  $\mathcal{D}_{x,w_2}$  (rather than two copies of  $\mathcal{D}_{x,w_1}$ ) can only be negligibly more than  $\epsilon/2$ . Hence for any inverse polynomial  $\epsilon$ , this error probability is bounded by  $\epsilon$ , as required.

Finally, we address the fact the p-UWM mapping algorithm above was defined to be randomized, to carry out the implementation of ( $N$  samples from)  $\mathcal{D}_{x,w}$ , given  $(x, w)$ . Since the definition of pseudo-uniqueness involves only two runs of the mapping algorithm, it is enough to include a pairwise independent hash function in the CRS which would be used to derive the required amount of randomness as a function of its input  $(x, w)$ .

## 6 Mildly Compact WM Implies NIZK

In this section, we show that a mildly compact WM - i.e., a CWM with compactness level  $\alpha = \text{poly}(\kappa)$  for some fixed polynomial  $\text{poly}(\cdot)$ , independent of the statement size  $|x|$  or the witness size  $|w|$  implies the existence of NIZK argument system. As mentioned in the introduction, we generalize the recent work of [10] who constructed NIZKs from SNARG by replacing SNARG with CWM with the above compactness level. In particular, we show a construction of *hidden-bits generator with subset-dependent proofs* (SDP-HBG) from  $\alpha$ -CWM. This result, along with the compiler of [10] that transforms any NIZK in the hidden-bits model (HBM-NIZK) to NIZK in the CRS model using SDP-HBG implies a construction of NIZK from any  $\alpha$ -CWM as long as  $\alpha = \text{poly}(\kappa)$  as defined above. Following [10] we first provide the definition of SDP-HBG.

### 6.1 Hidden-Bits Generator with Subset-Dependent Proofs

Following [10], we recall the notion of hidden-bits generator with subset-dependent proofs (SDP-HBG).

**Definition 8 (SDP-HBG).** A hidden-bits generator with subset-dependent proofs (SDP-HBG) consists of four PPT algorithms ( $\text{HBG}^{\text{sdp}}.\text{Setup}$ ,  $\text{HBG}^{\text{sdp}}.\text{GenBits}$ ,  $\text{HBG}^{\text{sdp}}.\text{Prove}$ ,  $\text{HBG}^{\text{sdp}}.\text{Verify}$ ) defined as follows:

1.  $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$  : The setup algorithm takes the security parameter  $1^\kappa$  and the length parameter  $1^n$  as input, and outputs a CRS  $\text{crs}$ .
2.  $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$  : The bits generation algorithm takes a CRS  $\text{crs}$  as input, and outputs a string  $r \xleftarrow{\$} \{0, 1\}^n$  and a state  $\text{st}$ .
3.  $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$  : The proving algorithm takes a state  $\text{st}$  and a subset  $I \subseteq [n]$  as input, and outputs a proof  $\pi$ .
4.  $\text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi)$  : The verification algorithm takes a CRS  $\text{crs}$ , a subset  $I \subseteq [n]$ , a string  $r_I \in \{0, 1\}^{|I|}$  and a proof  $\pi$  as input, and outputs either 1 or 0 indicating acceptance or rejection respectively.

A SDP-HBG is required to satisfy the following properties:

- **Correctness.** For any natural number  $n$  and  $I \subseteq [n]$ , we have:

$$\Pr \left[ \begin{array}{l} \text{crs} \leftarrow \text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n); \\ \text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi) = 1 : (r, \text{st}) \leftarrow \text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs}); \\ \pi \leftarrow \text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I) \end{array} \right] = 1$$

- **Somewhat Computational Binding.** There exists a constant  $\gamma < 1$  such that (1) for any polynomial  $n = n(\kappa)$  and for all  $\text{crs} \leftarrow \text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$ , there exists a subset  $\mathcal{V}^{\text{crs}} \subseteq \{0, 1\}^n$  such that  $|\mathcal{V}^{\text{crs}}| \leq 2^{n^\gamma \text{poly}(\kappa)}$  holds, and (2) for any PPT adversary  $\mathcal{A}$ , we have:

$$\Pr_{\text{crs} \leftarrow \text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)} \left[ r_I \notin \mathcal{V}_I^{\text{crs}} \wedge \text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi) = 1 : (I, r_I, \pi) \leftarrow \mathcal{A}(\text{crs}) \right] = \text{negl}(\kappa).$$

where  $\mathcal{V}_I^{\text{crs}} = \{r_I : r \in \mathcal{V}^{\text{crs}}\}$

- **Computational Hiding.** For any polynomial  $n = n(\kappa)$ ,  $I \subseteq [n]$ , any PPT adversary  $\mathcal{A}$ , we have:

$$\left| \Pr [\mathcal{A}(\text{crs}, I, r_I, \pi, r_I) = 1] - \Pr [\mathcal{A}(\text{crs}, I, r_I, \pi, r'_I) = 1] \right|$$

where  $r'_I$  denotes the substring of  $r$  on the positions corresponding to  $\bar{I} = [n] \setminus I$ .

A SDP-HBG is a weaker primitive than HBG, and [10] showed how to construct a SDP-HBG generically starting from any HBG.

## 6.2 Construction of SDP-HBG

In this section, we show how to construct a SDP-HBG from CWM and OWF. Our construction requires the following ingredients:

- An  $\lambda$ -entropic leakage-resilient weak PRF ( $\lambda$ -LR-wPRF)  $\mathcal{F} = \{F_K : \{0, 1\}^m \rightarrow \{0, 1\}^k\}_{K \in \{0, 1\}^k}$  (see Sect. 2.2.3), with key length  $k = k(\kappa, \lambda) = \lambda \cdot \text{poly}(\kappa)$ , input length  $m = m(\kappa, \lambda) = \lambda \cdot \text{poly}(\kappa)$ , and output length 1 bit. Here  $\lambda$  denotes the leakage parameter of the ELR-wPRF  $\mathcal{F}$ .
- A  $\alpha$ -CWM  $\text{CWM} = (\text{setup}, \text{map}, \text{check})$  for  $\alpha = \text{poly}(\kappa)$  for an arbitrary polynomial  $\text{poly}(\kappa)$  (independent of the length of the statement or witness) for the language  $L$  associated with the following relation  $R$ :

$$\left( (k', \{x_i\}_{i \in [k']}, \{r_i\}_{i \in [k']}, K) \right) \in R \iff r_i = F_K(x_i) \quad \forall i \in [k']$$

We now proceed to describe our construction.

1.  $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$  : Do the following:
  - (a) Run  $\mathbf{K} \leftarrow \text{setup}(\kappa, \ell)$ , where  $\ell$  is the length of the statement mentioned in the relation above.
  - (b) For all  $i \in [n]$ , sample  $x_i \xleftarrow{\$} \{0, 1\}^m$ .  
Return  $\text{crs} = (\mathbf{K}, \{x_i\}_{i \in [n]})$ .
2.  $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$  : Do the following:
  - (a) Parse the CRS as  $\text{crs} = (\mathbf{K}, \{x_i\}_{i \in [n]})$ .
  - (b) Sample key  $K \xleftarrow{\$} \{0, 1\}^k$ , and compute  $r_i = F_K(x_i)$  for all  $i \in [n]$ .  
Return  $(r = \{r_i\}_{i \in [n]}, \text{st} = (\text{crs}, K, r))$ .
3.  $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$  : Parse  $\text{st} = (\text{crs}, K, r)$  and  $\text{crs} = ((\mathbf{K}, \{x_i\}_{i \in [n]}))$ . Then compute  $w^* \leftarrow \text{map}(\mathbf{K}, (|I|, x_I, r_I), K)$  and return  $\pi := w^*$ .
4.  $\text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi)$  : Parse  $\text{crs} = ((\mathbf{K}, \{x_i\}_{i \in [n]}))$  and  $\pi$  as  $w^*$ .  
Return the output  $\text{check}(\mathbf{K}, (|I|, x_I, r_I), w^*)$ .

**Theorem 4.** *Let  $\kappa$  be the security parameter. If there exist a  $\lambda$ -entropic leakage-resilient weak PRF and a  $\alpha$ -CWM for all NP languages for  $\alpha = \text{poly}(\kappa)$  (where  $\text{poly}(\kappa)$  is some polynomial) such that  $\alpha \leq \lambda$  and that satisfies adaptive soundness, then there exists an SDP-HBG that satisfies somewhat computational binding and computational hiding.*

The detailed proof of this theorem is presented in the full version of the paper. The intuition behind the proof is given in the technical overview.

## 7 UWM Implies Deterministic-Prover NIZK

In this section we show that UWM implies *deterministic-prover* NIZK argument system (DP-NIZK) satisfying *perfect* zero-knowledge. Before this, we knew how to construct such a DP-NIZK argument system only from  $i\mathcal{O}$ . Below we briefly

sketch the construction and defer to the full version of the paper for the details of the construction and proof.

A DP-NIZK argument system is NIZK argument system where the prover and verifier are both *deterministic*. Apart from completeness and soundness we require perfect zero-knowledge property to hold, i.e., the simulated proofs are identically distributed to the real proofs. The setup algorithm of our DP-NIZK chooses a random seed  $z$  for a length-doubling pseudorandom generator  $G$  and sets  $y = G(z)$ . The CRS  $\text{crs}$  of DP-NIZK consists of the CRS  $\text{K}$  of UWM and the value  $y$ . The prover of DP-NIZK on input some  $(x, w)$  in the relation runs the UWM prover to get a representative witness  $w^*$  for the statement  $\hat{x}$  stating that “either  $x$  is true or  $y$  is pseudorandom”, using  $w$  as the witness. Note that the prover is deterministic. In the proof of soundness, we sample  $y$  uniformly at random, so that  $y$  is not in the image of  $G$ , except with negligible probability. At this point the soundness of DP-NIZK follows from the soundness of UWM. To prove ZK, the simulator uses the witness  $z$  to simulate the proofs. The uniqueness property of UWM guarantees that proofs computed by either of the witnesses result in the same proof. Hence, the zero-knowledge property follows.

## 8 Perfectly Sound Verifier UWM Implies $\mathbf{NP} = \mathbf{UP}$

In this section, we show that if a perfect sound verifier unique witness map (VUWM) exists (see Definition 6) then the complexity class  $\mathbf{NP}$  will be equal to the complexity class  $\mathbf{UP}$ , where  $\mathbf{UP}$  stands for unambiguous non-deterministic polynomial-time. Informally, the class  $\mathbf{UP}$  is the complexity class of decision problems solvable in polynomial time on an *unambiguous* Turing machine with at most *one* accepting path for each input. Hence it is easy to see that  $\mathbf{UP}$  contains the class  $\mathbf{P}$  and is contained in  $\mathbf{NP}$ . In the following we shall prove that  $\mathbf{NP} \subseteq \mathbf{UP}$ , assuming perfect sound VUWM. Let us first formally define the class  $\mathbf{UP}$ .

**Definition 9 (Complexity class  $\mathbf{UP}$ ).** *A language  $L \in \mathbf{UP}$  if there exists a two-input polynomial-time algorithm  $R$  and a constant  $c$  such that*

- *If  $x \in L$ , then there exists a unique certificate  $w$  with  $|w| = O(|x|)^c$  such that  $R(x, w) = 1$ .*
- *If  $x \notin L$ , there is no certificate  $w$  with  $|w| = O(|x|)^c$  such that  $R(x, w) = 1$ .*

Valiant and Vazirani [16] showed that  $\mathbf{NP} \subseteq \mathbf{RP}^{\text{promise-UP}}$ , which means that that there is a randomized reduction from any problem in  $\mathbf{NP}$  to a problem in  $\mathbf{Promise-UP}$ .

**Theorem 5.** *If perfectly-sound VUWM exists for an  $\mathbf{NP}$  relation  $R$ , then  $L_R \in \mathbf{UP}$ . In particular, if perfectly-sound VUWM exists for every  $\mathbf{NP}$  relation, then  $\mathbf{NP} = \mathbf{UP}$ .*

The proof of this theorem is presented in the full version of the paper.

## References

1. Beigel, R.: On the relativized power of additional accepting paths. In: Proceedings: Fourth Annual Structure in Complexity Theory Conference, University of Oregon, Eugene, Oregon, USA, 19–22 June 1989, pp. 216–224. IEEE Computer Society (1989)
2. Beigel, R., Buhrman, H., Fortnow, L.: NP might not be as easy as detecting unique solutions. In: Scott Vitter, J. (ed.) Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, 23–26 May 1998, pp. 203–208. ACM (1998)
3. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20465-4\\_7](https://doi.org/10.1007/978-3-642-20465-4_7)
4. Chakraborty, S., Prabhakaran, M., Wichs, D.: Witness maps and applications. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. Part I, volume 12110 of LNCS, pp. 220–246. Springer, Heidelberg (2020). [https://doi.org/10.1007/978-3-030-45374-9\\_8](https://doi.org/10.1007/978-3-030-45374-9_8)
5. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: 51st FOCS, pp. 511–520. IEEE Computer Society Press (2010)
6. Gat, E., Goldwasser, S.: Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex* **TR11**, 136 (2011)
7. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC, pp. 25–32. ACM Press (1989)
8. Goldwasser, S., Grossman, O., Holden, D.: Pseudo-deterministic proofs. In: Karlin, A.R. (ed.) ITCS 2018, volume 94 of LIPIcs, pp. 1–18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018)
9. 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](https://doi.org/10.1007/978-3-642-38348-9_10)
10. Kitagawa, F., Matsuda, T., Yamakawa, T.: NIZK from SNARG. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. Part I, volume 12550 of LNCS, pp. 567–595. Springer, Heidelberg (2020). [https://doi.org/10.1007/978-3-030-64375-1\\_20](https://doi.org/10.1007/978-3-030-64375-1_20)
11. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. Part II, volume 9615 of LNCS, pp. 447–462. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49387-8\\_17](https://doi.org/10.1007/978-3-662-49387-8_17)
12. Liu, Y., Pass, R.: On one-way functions and Kolmogorov complexity. In: 61st FOCS, pp. 1243–1254. IEEE Computer Society Press (2020)
13. Rackoff, C.: Relativized questions involving probabilistic algorithms. *J. ACM* **29**(1), 261–268 (1982)
14. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC, pp. 475–484. ACM Press (2014)
15. Valiant, L.G.: Relative complexity of checking and evaluating. *Inf. Process. Lett.* **5**(1), 20–23 (1976)



16. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. *Theor. Comput. Sci.* **47**(3), 85–93 (1986)
17. Zhandry, M.: How to avoid obfuscation using witness PRFs. In: Kushilevitz, E., Malkin, T. (eds.) *TCC 2016-A. Part II*, volume 9563 of LNCS, pp. 421–448. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49099-0\\_16](https://doi.org/10.1007/978-3-662-49099-0_16)



# Structure-Preserving Compilers from New Notions of Obfuscations

Matteo Campanelli<sup>1</sup>, Danilo Francati<sup>2</sup>(✉), and Claudio Orlandi<sup>2</sup>

<sup>1</sup> Protocol Labs, San Francisco, USA  
matteo@protocol.ai

<sup>2</sup> Aarhus University, Aarhus, Denmark  
{dfrancati,orlandi}@cs.au.dk

**Abstract.** The dream of software obfuscation is to take programs, *as they are*, and then generically compile them into obfuscated versions that hide their secret inner workings. In this work we investigate notions of obfuscations weaker than virtual black-box (VBB) but which still allow obfuscating cryptographic primitives preserving their original functionalities as much as possible.

In particular we propose two new notions of obfuscations, which we call *oracle-differing-input* obfuscation (odiO) and *oracle-indistinguishability* obfuscation (oiO). In a nutshell, odiO is a natural strengthening of *differing-input* obfuscation (diO) and allows obfuscating programs for which it is hard to find a *differing-input* when given only *oracle access* to the programs. An oiO obfuscator allows to obfuscate programs that are *hard to distinguish* when treated as oracles.

We then show applications of these notions, as well as positive and negative results around them. A few highlights include:

- Our new notions are weaker than VBB and stronger than diO.
- As it is the case for VBB, we show that there exist programs that cannot be obfuscated with odiO or oiO.
- Our new notions allow to generically compile several flavours of secret-key primitives (e.g., SKE, MAC, designated verifier NIZK) into their public-key equivalent (e.g., PKE, signatures, publicly verifiable NIZK) while preserving one of the algorithms of the original scheme (function-preserving), or the structure of their outputs (format-preserving).

## 1 Introduction

**Obfuscation and Its (Dream) Applications.** Obfuscation—the ability of running a program hiding its inner working—is a cryptographer’s dream. This is especially true of its most powerful instantiation, virtual black-box (VBB) obfuscation: anything a VBB-obfuscated program leaks can be simulated through oracle access to the function it computes [8]. It follows that one important application of VBB is to *generically transform* secret-key cryptographic primitives into their public-key counterparts (an approach sometimes referred to as *white-box cryptography*). For example, the seminal work of Diffie and Hellman [25]

already imagined compiling secret key encryption (SKE) into public key encryption (PKE) by letting the public key consist of the obfuscated encryption program  $\text{Enc}(k, \cdot)$ . Note that this compiler has the advantage of preserving the *format* of the underlying ciphertext, as well as the *function* used to perform decryption.

**Transforming Primitives, Nicely.** In this paper, we are interested in obfuscators that allow generic *structure preserving* transformation of large classes of cryptographic primitives, i.e., obfuscators that allow to compile cryptographic primitives while *preserving* parts of the original primitive. In particular, with the term structure-preserving, we refer to two main classes of transformations (from secret-key to public-key primitives), dubbed *function-preserving* and *format-preserving* transformations:

- *Function-preserving.* This first type of transformation does not alter the algorithms (one of which is then obfuscated during the transformation) of the secret-key primitive. An example of such a transformation is the one described by Diffie and Hellman, i.e., compile a SKE into a PKE by obfuscating (without any modification) the encryption algorithm and keep the decryption one unchanged.
- *Format-preserving.* This other type of transformation modifies the algorithms of the original secret-key primitive but it preserves the format of the output.<sup>1</sup> For example, in order to convert a SKE into a PKE, a format-preserving transformation may require to modify both (before obfuscating) the encryption and decryption algorithm. However, these modifications do not alter the format of ciphertexts (i.e., the ciphertexts of the resulting PKE is of the same format as the original SKE one).

We see this as an interesting design approach to transformation of primitives, worth of study of its own. Structure-preserving compilers are desirable because of: (i) *reusability/retrocompatibility* and (ii) *efficiency*. First, with function-preserving transformations we can reuse existing code, programs, libraries, constructions and their cryptanalysis. Cryptographic primitives deployed in hardware could reuse that same hardware for the transformed primitive, instead of having to be redesigned from scratch and possibly replaced in a production environment. Moreover, transformations that preserve the *format* of their output allow to reuse parsing-related software and to be retrocompatible with older standards (particularly important for legacy systems). Also, function- and format-preserving transformations maintain some of the scheme’s original efficiency guarantees such as preserving the running time of the (possibly heavily optimized) original function and its communication complexity, respectively.

**Nice Transformations from Weaker Obfuscation?** The seminal work in [7,8] has shown that the “dream version” of obfuscation, VBB, is in gen-

<sup>1</sup> Note that a function-preserving transformation is also format-preserving. This is because the former does not modify the algorithms of the original primitive. Hence, the format of the output is preserved by definition.

eral impossible, i.e., there exist programs that cannot be obfuscated through VBB. Since then cryptographers have defined new, weaker notions of obfuscations that could hopefully be constructed. One of the plausible weaker candidates in this sense is *indistinguishability obfuscation* (iO) that guarantees the indistinguishability of a pair obfuscated programs, only if the latter have the exact same input-output behavior. It is truly surprising that a notion of obfuscation as weak as iO has managed to generate so many applications [41]. However, most of the applications of iO are out of the spectrum of the “design once; obfuscate later”-approach that was dreamed in the beginning, i.e., generically compile an existing secret-key primitive that it has been designed without the intend of being obfuscated later in time. In fact, most iO-based constructions are quite involved and they are not generic since only carefully designed programs can be successfully obfuscated through iO. A clear example is [41] that leverages puncturable PRFs and pseudorandom generators (PRG) to build (from scratch) a SKE scheme that satisfies very specific properties (e.g., puncturability) which, in turn, allows iO to convert it into a PKE scheme. Intuitively, this is far from having a generic transformation since the SKE is built with the intent of being obfuscated through iO. It is therefore natural to ask the following question:

*Can we obtain generic structure-preserving transformations from notions of obfuscation weaker than VBB?*

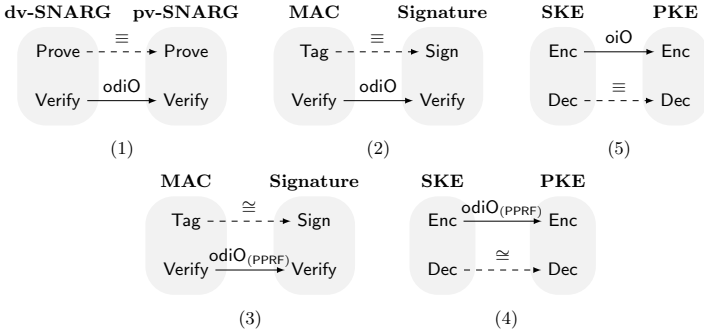
**Our Results: New Primitives, Compilers, Connections to Prior Notions.** In this work we propose two new definitions of obfuscation, *oracle-differing-input obfuscation* (odiO) and *oracle-indistinguishability obfuscation* (oiO), and apply them to structure-preserving transformations for several classes of primitives.

Recall that iO [8] only guarantees indistinguishability of obfuscations between pair of programs that have the exact same input/output behaviour. *Differing-input obfuscation* (diO) [1, 8, 17] is a stronger kind of obfuscation which guarantees the same indistinguishability property of iO but for pair of programs which might have different input/output behaviour, as long as it is computationally hard to find inputs on which the output of the programs differ, even when looking at the code of the programs. Our first notion, odiO, enriches the class of programs that can be securely obfuscated including any pair of programs for which it is hard to find differing-inputs, but when the distinguisher is given only oracle access to the programs. oiO then takes it a step further and allows to obfuscate any pair of programs that are indistinguishable when given as oracle.

In the paper we formally study the relationship between our new notions of obfuscation and the existing one. Note that:

$$\text{VBB} > \text{oiO} > \text{odiO} > \text{diO} > \text{iO}$$

meaning that a VBB-obfuscator is also an oiO-obfuscator, and so on. Intuitively, the separation are strict. Again, focusing only on the first inequality: while a VBB-obfuscator cannot leak anything about the program that cannot be learned



**Fig. 1.** The transformations (1)-(5) of this work: function-preserving on top row; format-preserving on bottom row. By odiO/oiO we denote an algorithm obtained through direct obfuscation of the one on the left; by  $\equiv$  one that is completely unchanged; by  $\cong$  one with minor changes but still able to take the same input; by (PPRF) we denote where we modify the algorithm through puncturable PRFs before obfuscation.

by the oracle version of the program, an oiO-obfuscator is allowed to leak any secret contained in its circuit, as long as these secrets do not allow to distinguish between the oracle programs. Focusing on oiO, odiO, diO, and iO, we have that all these notions provide the same flavor of security (i.e., two obfuscations are indistinguishable) but for different classes of circuits, each progressively contained into the other. For this reason, we have that  $oiO > odiO > diO > iO$ .

Note that odiO is stronger than diO. Hence, as considered by previous works for diO, this work assumes that current candidates of iO obfuscators are candidates obfuscators for odiO and oiO.

Still, since oiO and odiO are weaker than VBB, it is plausibly easier to build oiO and odiO obfuscators than VBB ones, at least for specific classes of programs: For example, it is known that point functions can be VBB-obfuscated, despite the general impossibility results for VBB; similarly, programs that differ in a single input (or polynomial number of inputs) can be diO-obfuscated, even if we believe that diO is unlikely to exist in general. In the same spirit, our results can be interpreted as showing that we can lift certain symmetric-key primitives to public-key primitives as long as specific functions (e.g., verification algorithms) can be odiO-obfuscated.

We then show that our new notions of obfuscation are enough for generic structure-preserving transformations of important cryptographic primitives. In particular we provide the following transformation (see also Fig. 1):

1. A *function-preserving* transformation from selectively sound succinct designated verifier non-interactive argument systems (dv-SNARG) into publicly verifiable ones (pv-SNARG) (Sect. 5.1); The same transformation allows transforming non-interactive argument systems that satisfy straight-line knowledge

- soundness, i.e., it is possible to extract (through a trapdoor) a valid witness from verifying proofs without interacting with the adversary;<sup>2</sup>
2. A *function-preserving* transformation from strong existentially unforgeable MACs into digital signatures that remains strongly unforgeable only in the presence of adversaries that can ask signatures of arbitrary messages in a selective fashion;
  3. A *format-preserving* transformation that leverages puncturable PRFs to convert selectively existentially unforgeable MACs into selectively existentially unforgeable digital signatures. In contrast to the previous (MACs to signatures) transformation, this is only format-preserving but achieves existential unforgeability under the standard notion of chosen message attacks (i.e., the adversary has adaptive oracle access to the signature algorithm);
  4. A *format-preserving* transformation that leverages puncturable PRFs to convert IV-based selectively secure SKEs into selectively IND-CPA secure PKEs. Here, IV-based SKEs refer to encryption schemes of the form  $\text{Enc}(k, m; iv) = (iv, c)$  where  $iv$  is the initialization vector (i.e., randomness) used to encrypt a message  $m$ . Note that most SKE used in practice are IV-based e.g., those based on block ciphers mode operations such as AES-CBC-mode, AES-CTR-mode, and so on.
  5. A *function-preserving* transformation from any semantically secure and key indistinguishable SKE into a selective IND-CPA PKE (Sect. 5.2). Here, the SKE's key indistinguishability property must hold under chosen message randomness attacks, i.e., it is infeasible to determine under which key a target message has been encrypted even if the adversary has oracle access to  $\text{Enc}(k, \cdot; \cdot)$  that accepts adversarially chosen messages and randomnesses.

Note that only the last transformation requires  $oiO$  (in order to use the key indistinguishability property of the SKE) whereas  $odiO$  is sufficient to achieve the other ones. Also, all the transformations that use puncturable PRFs are (only) *format-preserving*, i.e., the programs/algorithms of the compiled primitive are (slightly) modified but the format of the output is preserved.<sup>3</sup> We anticipate that all our transformations require  $odiO/oiO$  and they cannot be implemented using  $iO$  (or  $diO$ ). In a nutshell, this is because we focus on generic transformations from secret-key to public-key primitives. In order to be generic, we need to eventually reduce the security of the transformation to the security of the original secret-key primitive. If we wish to accomplish this reduction using either  $iO$  or  $diO$ , we need to put, into the obfuscated circuit, the key  $sk$  of the original (secret-key)

<sup>2</sup> As for straight-line knowledge soundness, we do not consider succinctness (i.e., we do not cover  $dv$ -SNARG/ $pv$ -SNARG) since, in order to have a straight-line extraction, the size of the proof is proportional to the size of the witness.

<sup>3</sup> We will elaborate on this later, but intuitively this is because the obfuscated program will use the puncturable PRF to generate a fresh symmetric key for different input (e.g., messages, initialization vectors). Hence, on decryption/verification, the receiver needs to evaluate the same PRF in order to recompute the symmetric key used to decrypt/verify a particular ciphertext/signature.

primitive. However, this is not possible. During the reduction  $sk$  is sampled and kept secret by the challenger. We provide more details in Sect. 1.1.

Although both  $odiO$  and  $oiO$  are weaker than  $VBB$ , this does not tell us anything about the plausibility of these new notions of obfuscation (and their applications). As a last contribution, we investigate whether  $VBB$ 's impossibility results of Barak et al. [7, 8] extends to either  $odiO$  or  $oiO$  (or both). In particular, Barak et al. [7, 8] shows the following two impossibility results regarding  $VBB$ . (i) The first states an *universal*  $VBB$  obfuscator does not exist, i.e. there exists (not necessarily natural) computations that cannot be obfuscated through  $VBB$ . (ii) The second states that even the specific applications/transformations of  $VBB$  we can naturally hope for are impossible (e.g., converting any SKE into a PKE).

What about the above and  $odiO/oiO$ ? We answer this question as follows:

- Result (i) does apply to  $odiO$  and  $oiO$ . This does not say much about how useful they are, so in the paper we explore result (ii) as well (see Sect. 6.1).
- Result (ii) applies only to transformation (5) for  $oiO$ . Moreover, we need to rework the original result (ii) from [7, 8] to extend it to transformation (5) since it does not apply as it is (see Sect. 6.2).
- Result (ii) does not apply at all to the applications/transformations we have for  $odiO$  and there seems no natural way to extend it to them. Hence, all our  $odiO$ -based transformations remain plausible.

Summing up, while the first result (i) applies to both our proposed notions,  $odiO$  is *not* at all subject to the second result (ii) (impossibility of applications), which is the most limiting one.

Expanding more about the above results, we provide two different negative results by adapting the techniques of [7, 8] to the case of  $odiO$  and  $oiO$ . First, we show that there exists an ensemble of circuits that neither  $odiO$  nor  $oiO$  cannot obfuscate, unconditionally (Sect. 6.1). Second, we show that the  $oiO$ -based function-preserving transformation (5) from any semantically secure and key indistinguishable SKEs into selective IND-CPA secure PKEs is inherently impossible (no matter what type of obfuscator is used to implement it).<sup>4</sup> We elaborate further on this in the technical overview (Sect. 1.1) and in the related Sect. 6.

**Why Study these New Notions if they are Still Subject to the [7, 8] Impossibility Results?** There are multiple responses to that (some of which we expand below):

1. The work of Barak et al. [7, 8] does not really say much about how useful  $odiO/oiO$  are (see also technical overview). Indeed, [7, 8] shows *two* types of impossibility results, i.e., impossibility of an universal obfuscator and impossibility of applications. As we mentioned above, these impossibilities do not

<sup>4</sup> Note that Barak et al. [8] demonstrates the impossibility of transforming a SKE into a PKE (through the obfuscation of its encryption algorithm  $Enc(k, \cdot)$ ) by building a (contrived) secure SKE that, after applying the transformation, yields an insecure PKE. However, their contrived SKE is not key indistinguishable. For this reason, in order to prove the impossibility of our  $oiO$ -based transformation (5) (from key indistinguishable SKE to PKE) we need to rework their result.

equally extend to both notions, e.g., impossibility of applications do not extend to our `odiO`-based transformations.

2. It is still important to study foundational aspects of obfuscation. We think ours are natural questions and natural notions to propose and to turn our attention to. As it is often the case in theoretical research, these notions may be connected to others in the future in unexpected ways. They might also motivate further work on notions that will turn out to be achievable (the [7, 8] prompted the quest for `iO` and other notions related to `VBB`).
3. Related to the above, both `odiO` and `oiO` might be useful for many “proof-of-concept” type of results that rely on `VBB`-obfuscation. In cases where these weaker definitions might suffice, the proposed notions could shed light on what security property is actually needed from the obfuscator to imply security of the overall construction.<sup>5</sup> As an example, our results can be interpreted as follows: If a particular circuit (e.g., secret-key verification/encryption algorithm) can be `odiO`-obfuscated (resp. `oiO`-obfuscated) then we can lift a particular secret-key primitive into its public-key flavor (e.g., MAC to signatures, SKE to PKE).
4. `VBB` and `odiO/oiO` are still distinct notions and with distinct flavors of security (simulation- vs indistinguishability-based). Moreover, nonetheless the known impossibility results, research on `VBB` is still active such as identifying specific and interesting class of circuits that can be securely `VBB`-obfuscated [33, 43, 45]. The same can be investigated for the case of `odiO/oiO`. For example, there could exist a specific class of circuits that can be `oiO/odiO`-obfuscated but not `VBB`-obfuscated. Or there could exist circuit classes that are `VBB`-obfuscatable, but that can still be `odiO/oiO`-obfuscated more efficiently or from significantly weaker assumptions.

Lastly, we stress that `odiO/oiO` may have other interesting applications. This work focuses on secret-key to public-key transformations since these are most prominent applications of `VBB` and, we believe that studying `odiO/oiO` in the same context provides a better understanding about the relations between `odiO/oiO` and `VBB`, including their limitations.

## 1.1 Technical Overview

**Oracle-Differing-Input Obfuscation (`odiO`).** The notion of `odiO` is a variant of the notion of differing-input obfuscation, or `diO`. What is common with `diO`, for example, is that: (i) we are given a sampler  $S$  that outputs two circuits  $C_0$  and  $C_1$  and some auxiliary information  $\alpha$ ; (ii) the output of the sampler should satisfy some property  $P$  (we call such sampler “permissible”); (iii) if the sampler satisfies property  $P$  then the obfuscated circuits  $\text{Obf}(C_0)$  and  $\text{Obf}(C_1)$  should look indistinguishable to a PPT adversary given also in input  $\alpha$ . Also, in both `diO` and `odiO`, the property  $P$  corresponds to “no PPT  $D$  can find a *differing*

<sup>5</sup> This follows the same spirit of the UCE framework proposed by Bellare et al. [9] that allows to identify which property of the random oracle model (ROM) is needed to imply security of the (ROM-based) construction..



input  $x$  for  $C_0$  and  $C_1$  (given in input  $\alpha$ )”, that is an input  $x$  such that  $C_0(x) \neq C_1(x)$ . Where the two definitions diverge is that in diO algorithm D takes as input the *actual representation* (the code) of the two circuits, whereas in odiO D only has *oracle access* to the functions computed by  $C_0$  and  $C_1$ .

An example of sampler that is permissible for odiO but not diO is the following: consider two programs  $C_0$  and  $C_1$  where their only (high-entropy) differing input is encoded as a comment in their code. Given their code it is easy to find such input, but not with oracle access to them. We provide more examples when we discuss our transformations below.

**Public-key “Forgery-based” Transformations through odiO.** We show that odiO is particularly suitable for transforming a general class of primitives—which we informally dub *forgery-based*—from their secret-key to their public-key version. By forgery-based we mean a primitive where the security is defined roughly as follows: “No adversary can produce (forge) a string passing a given test without knowledge of a certain secret (or if a certain condition does not hold)”. Straightforward examples of this type of primitives include message-authentication codes (MACs) and digital signature, but non-interactive proof systems and signatures of knowledge [24] also capture this intuition.

The properties of odiO are sufficient for compiling the forgery-based primitives (1)-(3) listed above. We now give the main intuitions behind our transformations and their security. Our goal is to transform a primitive allowing us to verify a string through knowledge of secret into one that can do the same without such knowledge. Let us denote the first generic verification algorithm by  $\text{Verify}(\text{sk}, \dots)$ ;<sup>6</sup> we aim to transform it into a public key equivalent  $\text{Verify}'(\text{pk}, \dots)$ . Our construction is straightforward: We define  $\text{pk}$  as the odiO-obfuscation of  $\text{Verify}(\text{sk}, \dots)$ , and the program  $\text{Verify}'(\text{pk}, \dots)$  simply runs the program encoded in  $\text{pk}$ .

We now argue that the above is secure in a selective-security-flavored setting. In general, in such a setting, the adversary first claims some input (e.g., a message or an NP statement) for which it would like to forge a valid string (e.g., a signature or a proof). The rest of the intuition is better conveyed being specific. We thus focus on the setting of non-adaptive (selective) security in non-interactive proof systems where the verifier has the syntax  $\text{Verify}(\text{vrs}, x, \pi)$  and  $\text{vrs}$  is the (secret) verification key,  $x$  is a public statement (allegedly in a language  $\mathcal{L}$ ),  $\pi$  is the proof. In this security game, for any input  $\hat{x} \notin \mathcal{L}$ , the adversary should not be able to forge a corresponding valid proof *after* seeing the public parameters (aka, common reference string or  $\text{crs}$ ). We now show how to reduce the security of the publicly verifiable construction to that of the original (designated verifier) one applying odiO security. Recall that the security property of odiO must refer to a given sampler returning pairs of circuits. We require that our odiO obfuscator is secure against a sampler that returns  $(C_0, C_1)$  (we ignore the auxiliary input here) where:

<sup>6</sup> The rest of the input besides the key is irrelevant for this discussion.

- $C_0$  takes as input  $x$  and  $\pi$  and returns  $\text{Verify}(\text{vrs}, x, \pi)$ .
- $C_1$  behaves like  $C_0$  *except* that it immediately returns 0 whenever  $x = \hat{x}$ .

The two circuits clearly satisfy the `odiO` permissibility notion since finding a differing input through oracle access to them would violate the original hypothesis of soundness (the only differing inputs are valid proofs for  $\hat{x}$ ).<sup>7</sup>

Thus we can move to an hybrid where the `crs` is an obfuscation of  $C_1$ , and indistinguishability of the hybrids follows from the security of the `odiO` obfuscator. But now note that by construction of  $C_1$ , when `crs` =  $\text{Obf}(C_1)$ , an adversary by definition cannot produce a valid for  $\hat{x}$ . Moreover, we obtain (for free) that our transformation preserves zero-knowledge since it is function-preserving and the `Prove` algorithm is not modified (see Remark 5.3).

The blueprint for the construction and security proof above can be adapted (with the appropriate care) to the other forgery-settings (2)-(3) for which we propose transformations. For transformation (2)—which yields selectively-secure strongly unforgeable signatures—one technical challenge is that we need to simulate the queries to the signing oracle. Since these queries are selective we can embed them in one of the circuits we obfuscate during the hybrid arguments. Transformation (3) requires additional care since it yields a signature scheme secure against an adversary with *adaptive* queries to the signing oracle. To do so we slightly modify the signature algorithm and use a (puncturable) PRF to generate a fresh one-time symmetric-key used to sign a single message. The verification algorithm is similarly adapted and then obfuscated. Due to the use of the PRF, the transformation is not function-preserving but only format-preserving.

**Compiling Extractable Argument Systems.** We are able to extend our result for argument schemes satisfying *soundness* to arguments that satisfy *knowledge soundness*. This is achieved by the exact same function-preserving construction from `odiO`.<sup>8</sup> We are able to compile an adaptively-secure straight-line extractable designated verifier argument into an adaptively-secure straight-line extractable publicly verifiable argument. Note that, when considering straight-line extractability, proofs are not succinct anymore; hence, in this case we cover `dv-NIZK` and `pv-NIZK`. In contrast to soundness—which achieves only selective security—here we are able to preserve adaptive security. Again, the

---

<sup>7</sup> In particular, soundness (of underlying designated-verifier non-interactive proof system) must hold even if the adversary has oracle access to the verification algorithm. The latter is essential during the reduction to simulate the input-output behavior of the two circuits (treated as oracles). Hence, our transformation does not apply to non-interactive proofs systems that suffer from the so called verifier rejection problem, i.e., giving oracle access to the verifier allows the adversary to break soundness..

<sup>8</sup> Despite the construction is the same, the sampler required to prove knowledge soundness is different.

transformation is function-preserving and it does not alter the Prove algorithm. Hence, zero-knowledge is preserved (see also Remark 5.3). To the best of our knowledge ours is the first work applying obfuscation in the context of extractability in proof schemes.

### Using **odiO** for Public-key Encryption through Puncturable PRFs.

So far we discussed how **odiO** is particularly useful for forgery-flavored primitives. We observe, however, that we are able to prove security of another type of primitive, encryption. In the full version of this work [21], we show how to compile **IV**-based selectively secure SKEs (whose ciphertexts have the form  $\text{Enc}(k, m; iv) = (iv, c)$ ) into selectively IND-CPA secure PKEs. Our obfuscated circuit (that will be our **pk**) uses two puncturable PRFs: The first to generate the initialization vector  $iv$  from the randomness given to the PKE's  $\text{Enc}$  and, the second to generate a one-time fresh symmetric-key (used to encrypt) from  $iv$ .<sup>9</sup> The decryption algorithm has access to the key for the second PRF and takes as input the ciphertext  $(iv, c)$ . It can then regenerate the key and thus decrypt. Note that this transformation is only format-preserving since we slightly modify both encryption and decryption algorithm to embed the evaluation of the PRF.

**Oracle-Indistinguishability Obfuscation (oiO).** The notion of **oiO** represents a natural strengthening of **odiO**. It has similar features to **diO** and **odiO** in that it requires samplers that output pairs of circuits satisfying some permissibility predicate  $P$ . While the permissibility predicate in **diO** and **odiO** requires hardness of finding a differing-input, in **oiO** we have a weaker permissibility predicate (which in turn makes **oiO** stronger than **odiO**): in **oiO** the sampler must output pairs of circuits such that an adversary (given also as input related auxiliary string  $\alpha$ ) cannot distinguish the circuits while having only oracle access to them. An example of a sampler that is permissible for **oiO** but not **odiO** is the one where  $C_0$  and  $C_1$  are both PRFs but with different keys, since they differ on (almost) every input but their output distributions are indistinguishable.

### Public-key “Indistinguishability-based” Transformations through **oiO**.

While **odiO** is suitable for transforming forgery-based primitives, **oiO** has synergies with indistinguishability-based primitives, i.e. where “No adversary can distinguish between two distributions without knowledge of a certain secret”. Natural examples are encryption schemes where the distributions to distinguish are the encryption of different messages (e.g., IND-CPA security).

---

<sup>9</sup> If, instead of generating  $iv$  using the first PRF, we allow the circuit to take directly in input  $iv$  then the PKE (output by the transformation) is trivially broken. This is because (following the syntax of the **IV**-based SKE)  $iv$  is included into the ciphertext. Hence, an adversary can break the selective IND-CPA security of the compiled PKE by simply re-encrypting a message using the  $iv$  that is included into the challenge ciphertext.

Through *oiO* we are able to prove the security of a more general transformation (compared to (4)) from SKEs to PKEs. Starting from a symmetric encryption algorithm  $\text{Enc}(k, \cdot; \cdot)$ , our aim is to transform it into something with the following syntax  $\text{Enc}(\text{pk}, \cdot; \cdot)$ , where  $\text{pk}$  is a public key. Our transformation is identical to the one proposed by Diffie and Hellman [25]: We define  $\text{pk}$  as the *oiO*-obfuscation of  $\text{Enc}(k, \cdot; \cdot)$  for some honestly chosen symmetric key  $k$ . To claim the IND-CPA security of the above transformation, we need to assume that the initial SKE is key indistinguishable under (adversarially) chosen message randomness attacks. The latter allows us to build a sampler that satisfies the permissibility predicate of *oiO*. In particular, the sampler returns  $(C_0, C_1)$  (again, we ignore the auxiliary input here) where:

- $C_0$  takes as input  $m$  and  $r$  and returns  $\text{Enc}(k, m; r)$ .
- $C_1$  is identical to the above except that it uses a different (honestly generated) symmetric key  $k'$ .

Intuitively, the circuits satisfy the *oiO* permissibility notion since any adversary that is able to distinguish between oracles  $C_0$  and  $C_1$  would also violate the key indistinguishability security of the SKE. Now, since the obfuscations of these two circuits are indistinguishable, we can reduce the security of the PKE to the security of the original SKE. Consider the standard IND-CPA experiment of PKE where  $\text{pk}$  is set to the obfuscation of  $C_0$  and the challenge ciphertext  $c$  is computed as  $c = \text{pk}(m_b; r) = \text{Enc}(k, m_b; r)$  for  $r$  randomly chosen. We can now do a hybrid where  $\text{pk}$  is set to the obfuscation of  $C_1$  whereas the challenge ciphertext is still computed as  $c = \text{Enc}(k, m_b; r)$  where  $k$  is the key hardcoded in  $C_0$ . Since the ciphertext  $c$  is computed using a key  $k$  that is not the obfuscated one (recall  $C_1$  uses an independent key  $k'$ ), we can now conclude the proof by doing a reduction to the semantic security of the original SKE. We highlight that this proof technique works only if we consider selective IND-CPA security. This is because the sampler needs to output an auxiliary input that is an honest encryption of  $m_b$  under the key  $k$  (hardcoded into  $C_0$ ). This is fundamental to simulate the challenge ciphertext (of the selective IND-CPA experiment) and concludes the hybrid argument.

**Why aren't *diO*/*iO* Sufficient for these Transformations Above?** We observe that each of the compilation described above would not be feasible with either *iO* or *diO*. Intuitively, this is because we would eventually need to reduce the security of our transformations (*pv*-SNARG, signature, PKE) to the security of the original secret-key primitive (*dv*-SNARG, MAC, SKE). However, in the latter experiment the secret-key  $\text{sk}$  (e.g., a *vrs* or a symmetric-key), that we need to obfuscate in order to conclude the reduction, is sampled and kept secret by the challenger. This makes *iO* and *diO* insufficient since we are not able to satisfy their permissibility notion during this reduction. For the case of *iO*, during the reduction, the only thing we could do is to obfuscate different circuit  $C_1$  that does not use the secret-key  $\text{sk}$  sampled by the challenger. However, this  $C_1$  will have (with overwhelming probability) a different input/output behavior compared to  $C_0$  (the original obfuscated circuit of the transformation that, in turn, contains  $\text{sk}$ ).

A similar discussion applies to diO. For the sake of concreteness, consider transforming a dv-SNARG into a pv-SNARG by publishing an obfuscation of the circuit  $C_0$  which implements the dv-SNARG verification algorithm using an hard-coded verification key  $\text{vrs}$ . During the reduction to the security of the underlying scheme we are not allowed to use the secret verification key  $\text{vrs}$ . Thus, during the reduction, we can only move to a hybrid where we obfuscate a circuit  $C_1$  that does not use the  $\text{vrs}$ . But then we cannot argue that it is hard to find differing-inputs for  $C_0, C_1$ . In this specific case, the distinguisher could simply produce proofs  $\pi$  for true statements  $x$  and submit them to the circuits. While  $C_0$  (using the  $\text{vrs}$ ) returns 1,  $C_1$  (without the  $\text{vrs}$ ) is unable to verify the proof and cannot return a consistent output. Similar arguments apply to the other transformations.

**The Landscape of Limitations of odiO/oiO.** The seminal work of [7,8] explores the boundaries of obfuscation in several directions. As it is well known they show that there are (not necessarily natural) computations which are impossible to obfuscate using VBB. Moreover, [7,8] also shows that VBB-obfuscation cannot be used for securely performing certain structure-preserving transformations. In this direction, they show a (contrived but secure) SKE that turns into an insecure PKE scheme when compiled using obfuscation. We show that the results of [7,8] can be extended to the setting of odiO and oiO. In particular, we show that there (unconditionally) exist samplers that are odiO/oiO permissible but are not obfuscatable. Specifically we sample (somewhat contrived) circuits  $C_s$  with an embedded secret  $s$  that remains “hidden enough” when only oracle access is allowed (thus being odiO/oiO permissible). We then show that, once given access to the obfuscated circuit, it becomes possible to “partially extract” this secret  $s$ . Finally, we show that (since this sampler cannot be obfuscated) our oiO-based transformation (5) (from semantically secure and key indistinguishable SKE to selectively IND-CPA PKE) is inherently impossible, regardless of the strength of the obfuscator used. This is done by using the unobfuscatable circuits to build a contrived SKE (satisfying semantic security and key indistinguishability) that, once compiled, yields an insecure PKE. As mentioned, a similar impossibility result was given in [8, Theorem 4.10]. However their contrived SKE does not satisfy key indistinguishability and, for this reason, it cannot be directly used to show the infeasibility of our transformation (5). Thus, our negative result strengthens the one of [8] since ours apply to a smaller class of SKEs (i.e., SKEs with stronger notions of security) that satisfy key indistinguishability under chosen message randomness attacks. Note that while we just argued that the oiO-based transformation in (5) is inherently impossible, our odiO-based transformations (1)-(4) remain plausible as the impossibility results do not seem to extend. We elaborate further in Sect. 6.2.

## 1.2 Future Directions

Our work opens up several interesting future directions. How to generally formalize structure-preserving transformations? Can we characterize what type of games can be transformed (from “secret” to “public” key) through `odiO`? Several, but not all those we achieve, seem to have a “forgery” flavor to them (MAC, NIZKs, etc.). What are further connections between our proposed notions of obfuscation and VBB, `iO` and `diO`? While the techniques in [8] seem to fail to show that some of our transformations are paradoxical, what are other techniques that could shed light on further limitations of `odiO` `oiO`? Can we leverage our techniques for going from secret-key to public-key variants of different cryptographic primitives than those we consider here, e.g., proofs of retrievability [42]?

## 2 Related Work

Barak et al. [7,8] investigate the feasibility of obfuscation. They focus on virtual black-box (VBB) obfuscation, where an obfuscated program/circuit should leak no information except for its input-output behaviour. They show: 1) that a general VBB obfuscator cannot exist since there are circuits that cannot be unconditionally obfuscated in the VBB paradigm; 2) that most of the intriguing applications of VBB are impossible (including the suggestion of Diffie and Hellman’s of building a PKE by obfuscating the SKE encryption algorithm with an embedded symmetric key). On the positive side, several works have shown that some restricted classes of circuits can be securely VBB-obfuscated [23,33,43,45]. Goldwasser and Kalai [30,31] and Bitansky et al. [13] extended VBB’s impossibility results to the case of auxiliary information demonstrating that other “natural” circuits cannot be VBB obfuscated when some (dependent or independent) auxiliary information are available. In addition, [13] demonstrated that the availability of auxiliary information is equivalent to VBB with universal simulation. Goldwasser and Rothblum [32] proposed the notion of best-possible obfuscation that guarantees that the obfuscation of a circuit leaks as little information as any other circuit implementing the same functionality. They show that a separation between VBB and best-possible obfuscation and an impossibility result (for both) in the random oracle model. Other works [6,20,22,37,38,40] studied the (in)feasibility of VBB in different idealized models.

To avoid the VBB paradigm (and its impossibility results), [8] suggested two weaker security definitions of obfuscation: indistinguishability obfuscation (`iO`) and differing-input obfuscation (`diO`). The former has obtained a lot of interest thanks to its applications, as initially shown by Sahai and Waters [41]. The first work that proposed a candidate `iO` construction is by Garg et al. [26] that built `iO` via multilinear maps. Subsequent works [2–4,15,16,19,28,29,36,39] focused on both the relations of `iO` and other primitives (e.g., functional encryption) and new candidates construction from weaker assumptions. These works led to the recent works of Jain et al. [35] and Wee and Wichs [44]. [35] built (sub-exponentially secure) `iO` from the sub-exponential hardness of LWE, learning parity with noise, and boolean pseudorandom generators in  $NC^0$ . On the other

hand, [44] proposed the first construction based solely on lattices and LWE. Their construction relies on a new falsifiable LWE assumption.

As for  $\text{diO}$ , [1, 10, 17, 17] proposed different formalization of  $\text{diO}$  (for both circuits and Turing machines) and showed different applications. On the negative side, [11, 18, 27] showed that, in the presence of (some) auxiliary information (e.g., samplers), a general  $\text{diO}$  obfuscator may not exist. Notably, Bellare et al. [11] showed that if sub-exponentially secure one-way functions exist then a sub-exponentially secure general  $\text{diO}$  obfuscator for Turing machines does not exist, i.e., there exists a sampler that outputs two Turing machines and some auxiliary information that cannot be obfuscated through  $\text{diO}$ . Moreover, they show that the impossibility result extends to  $\text{diO}$  for circuits, if SNARKs exist. Garg et al. [27] showed a similar result for  $\text{diO}$  for circuits under the conjecture that a special-purpose obfuscator exists (i.e., an obfuscator that does not follow from  $\text{diO}$ ). All the negative results of [11, 18, 27] rely on the fact that the sampler can silently provide a trapdoor that allows an adversary to distinguish between two obfuscations whereas the trapdoor does not help in finding a differing-input. Because of this, Ishai et al. [34] proposed the weaker notion of public-coin  $\text{diO}$  where the random coins of the sampler are public, i.e., a sampler cannot hide any trapdoor in the auxiliary information.

Among weaker notions of obfuscation, we also find virtual gray-box obfuscation (VGB) [12, 14]. This notion is close to that of VBB but models the simulator as semi-bounded, i.e., unbounded in running time but limited to a polynomial number of oracle queries. VGB is equivalent to another notion, strong  $\text{iO}$  ( $\text{siO}$ ), where it holds that  $\text{Obf}(C_0) \approx_c \text{Obf}(C_1)$  whenever the pair  $(C_0, C_1)$  is sampled from a concentrated distribution  $\mathbf{D}$ : For every input  $x$ , the probability that  $C_0(x)$  and  $C_1(x)$  do not return to common output  $\text{maj}_{\mathbf{D}}(x)$  is negligible (where  $\text{maj}_{\mathbf{D}}(\cdot)$  is defined with respect to the concentrated distribution  $\mathbf{D}$  taken into account). Observe that concentrated distributions are a generalization of evasive functions [5]. Intuitively,  $\text{siO}$  is weaker than  $\text{odiO}$  (and  $\text{oiO}$ ) since circuits (sampled from concentrated distributions) are oracle-diffing-input even against semi-bounded adversaries. Also, note that  $\text{siO}$  is not powerful enough to achieve structure-preserving transformations. Intuitively, because  $\text{siO}$  is able to obfuscate distributions of circuits that “pass” an information theoretical test. This is an obstacle when trying to implement our structure-preserving transformations since our objective is to compile/obfuscate primitives whose security follows from computational assumptions.

### 3 Preliminaries on Obfuscation

We assume the reader to be familiar with standard cryptographic notation and definitions. Our notation and all the standard definitions used in the paper can be found in the full version.

**Indistinguishability Obfuscation and Differing-input Obfuscation.** Let  $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$  be an ensemble of functionally equivalent circuits (of same size),

i.e.,  $\forall \lambda \in \mathbb{N}, \forall C_0, C_1 \in \mathcal{C}_\lambda, \forall x \in \{0, 1\}^{\ell_{in}}, C_0(x) = C_1(x)$  and  $|C_0| = |C_1|$ . Indistinguishability obfuscation (iO) [7] guarantees that the obfuscation of any two functionally equivalent circuits  $C_0, C_1 \in \mathcal{C}_\lambda$  are computationally indistinguishable. The stronger notion of differing-input obfuscation (diO) [1, 8, 17] considers the larger class of differing-input circuits, i.e., circuits that differ on hard to find inputs. Below, we introduce the definition of diO with respect to samplers responsible of sampling two differing-input circuits and (some) auxiliary information.

**Definition 3.1.** *A sampler  $S$  for an ensemble of circuits  $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$  is a PPT algorithm that, on input the security parameter  $1^\lambda$ , it outputs two circuits  $C_0, C_1 \in \mathcal{C}_\lambda$  such that  $|C_0| = |C_1|$  and (possibly) some auxiliary information  $\alpha$ .*

**Definition 3.2.** (diO-sampler) *We say a sampler  $S$  (Definition 3.1) is a diO-sampler if for every PPT adversary  $A$  we have*

$$\mathbb{P} \left[ C_0(x) \neq C_1(x) \mid (C_0, C_1, \alpha) \leftarrow S(1^\lambda), x \leftarrow A(1^\lambda, C_0, C_1, \alpha) \right] \leq \text{negl}(\lambda).$$

**Definition 3.3.** [Differing-input obfuscation] *Let  $\mathcal{S}$  be an ensemble of diO-samplers (Definition 3.2). For every  $S \in \mathcal{S}$ , let  $\mathcal{C}^S = \{\mathcal{C}_\lambda^S\}_{\lambda \in \mathbb{N}}$  be the ensemble of circuits output by  $S$ . A PPT algorithm  $\text{Obf}$  is a  $(\mathcal{S})$ -diO-obfuscator for the ensemble  $\mathcal{S}$  if the following conditions are satisfied:*

**Correctness.**  $\forall S \in \mathcal{S}, \forall \lambda \in \mathbb{N}, \forall C \in \mathcal{C}_\lambda^S, \forall x \in \{0, 1\}^{\ell_{in}},$  we have  $C'(x) = C(x)$  where  $C' \leftarrow S(\text{Obf}(1^\lambda, C))$ .

**Polynomial slowdown.** *There exists a polynomial  $p$  such that  $\forall S \in \mathcal{S}, \forall C \in \mathcal{C}_\lambda^S,$  we have  $|\text{Obf}(1^\lambda, C)| \leq p(|C|)$ .*

**Indistinguishability.** *For every  $S \in \mathcal{S}$ , every PPT adversary  $D$ , we have that*

$$\left| \mathbb{P} [D(1^\lambda, \text{Obf}(1^\lambda, C_0), \alpha) = 1] - \mathbb{P} [D(1^\lambda, \text{Obf}(1^\lambda, C_1), \alpha) = 1] \right| \leq \text{negl}(\lambda),$$

where  $(C_0, C_1, \alpha) \leftarrow S(1^\lambda)$ .

The above definition is parametrized by an ensemble of diO-samplers since some negative results for diO are known [11, 27] (see next). Because of this, an *universal* (general) diO-obfuscator may not exist, i.e., a diO-obfuscator that obfuscates any diO-sampler.

**Negative Results.** In the setting of Turing machines (not covered by this paper), Bellare et al. [11] show that if sub-exponentially secure one-way functions exist then a sub-exponentially secure diO-obfuscator  $\text{Obf}$  for any sampler for Turing machines does not exist (i.e., there exists a particular sampler that cannot be diO-obfuscated). We stress that the main impossibility result covers Turing machines but, as described by [11], if SNARKs exist the negative result can be extended to diO for circuits. Garg et al. [27] show that under the conjecture that a special-purpose obfuscator exists (i.e., an obfuscator that does not follow from the existence of a diO-obfuscator) then a diO-obfuscator  $\text{Obf}$  for any sampler for circuits does not exist. We highlight that both [11, 27] show that only



“some” diO-samplers cannot be obfuscated. Indeed, both works rely on samplers that output complex auxiliary information  $\alpha$  ( $\alpha$  is itself an obfuscation of contrived circuit/Turing machine). Hence, this does not rule out the possibility of obfuscating the same class of circuits/Turing machines under simpler auxiliary information.

**Virtual Black-box Obfuscation.** Virtual black-box obfuscation (VBB) [7], is the strongest known notion of obfuscation. In a nutshell, a VBB-obfuscator guarantees that having an obfuscation of a circuit  $C$  is “equivalent” to having oracle access to  $C$ . We consider the weakest notion of VBB that requires the adversary (and the simulator) to output a single bit. This is equivalent to asking the adversary/simulator to compute/determine an arbitrary predicate  $\pi(C)$  of the original circuit [7]. Similarly to diO, we consider VBB with respect to samplers responsible to sample a circuit and (some) auxiliary information. This will allow us to provide a meaningful comparison between VBB and diO, odiO, oiO.

**Definition 3.4.** (VBB-sampler) A VBB-sampler  $S$  for an ensemble of circuits  $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$  is a PPT algorithm that, on input the security parameter  $1^\lambda$ , it outputs a circuit  $C \in \mathcal{C}_\lambda$  and some auxiliary information  $\alpha$ .

**Definition 3.5 (Virtual black-box obfuscation).** Let  $\mathcal{S}$  be an ensemble of VBB-samplers (Definition 3.4). For every  $S \in \mathcal{S}$ , let  $\mathcal{C}^S = \{\mathcal{C}_\lambda^S\}_{\lambda \in \mathbb{N}}$  be the ensemble of circuits output by  $S$ . A PPT algorithm  $\text{Obf}$  is a  $(\mathcal{S})$ -VBB-obfuscator for the ensemble  $\mathcal{S}$  if the following conditions are satisfied:

**Correctness.**  $\forall S \in \mathcal{S}, \forall \lambda \in \mathbb{N}, \forall C \in \mathcal{C}_\lambda^S, \forall x \in \{0, 1\}^{\ell_{in}},$  we have  $C'(x) = C(x)$  where  $C' \leftarrow^* \text{Obf}(1^\lambda, C)$ .

**Polynomial slowdown.** There exists a polynomial  $p$  such that  $\forall S \in \mathcal{S}, \forall C \in \mathcal{C}_\lambda,$  we have  $|\text{Obf}(1^\lambda, C)| \leq p(|C|)$ .

**Virtual black-box simulation.** For every PPT adversary  $A$ , there exists a PPT simulator  $\text{Sim}$  such that for every  $S \in \mathcal{S}$ , we have

$$\left| \mathbb{P} [A(1^\lambda, \text{Obf}(1^\lambda, C), \alpha) = 1] - \mathbb{P} [\text{Sim}^{C(\cdot)}(1^\lambda, 1^{|C|}, \alpha) = 1] \right| \leq \text{negl}(\lambda),$$

where  $(C, \alpha) \leftarrow^* S(1^\lambda)$ .

Note that VBB is a much stronger flavor of obfuscation than diO and iO for two reasons. First, VBB defines the concept of ideal/oracle obfuscation, i.e., an obfuscated circuit behaves as an oracle. Second, VBB is a simulation-based definition (whereas both iO and diO are indistinguishability-based), i.e., any bit of leakage (that can be retrieved from the obfuscation of a circuit) can be simulated (except with negligible probability) having only oracle access to the unobfuscated circuit.

**Impossibility Results.** VBB is a very interesting notion of obfuscation since it has several important applications (e.g., it permits to convert a SKE into PKE). However, VBB-obfuscation turned out to be impossible for several and reasonably simple class of circuits/samplers [7, 8, 13]. Moreover, also several applications of

VBB are impossible to achieve. As an example, Barak et al. [8, Theorem 4.10] have shown that there exist a SKE that cannot be transformed into a PKE by (simply) obfuscating the SKE’s encryption algorithm (a similar impossibility result applies also to PRFs, MACs, and signatures). Still, VBB-obfuscation is still possible for other class of circuits/samplers. Examples are compute-and-compare programs [45] (also known as lockable obfuscation [33]) and point functions [43].

## 4 Oracle-Differing-Input and Oracle-Indistinguishability Obfuscation

In this section, we propose two new notions of obfuscation, dubbed *oracle-differing-input obfuscation* and *oracle-indistinguishability obfuscation* (odiO and oiO in short). Both odiO and oiO are the result of two natural extensions of diO (resp. iO): they introduce the notion of oracle circuits (as in VBB) while keeping the indistinguishability property of diO (resp. iO). In a nutshell, odiO requires that the obfuscations of two circuits  $C_0, C_1$  are computationally indistinguishable if the latter two are differing-input circuits when treated as oracles, i.e., an adversary cannot find an input  $x$  such that  $C_0(x) \neq C_1(x)$  when given oracle access to both  $C_0$  and  $C_1$ . On the other hand, oiO provides the same indistinguishability guarantee with respect to circuits  $C_0, C_1$  that are computationally indistinguishable when treated as oracles.

As usual, we define odiO and oiO with respect to an ensemble of samplers responsible of generating the circuits  $C_0, C_1$  and (possibly) some auxiliary information  $\alpha$ .

**Definition 4.1.** (*odiO- and oiO-sampler*) *Let  $\text{type} \in \{\text{odiO}, \text{oiO}\}$ . We say a sampler  $S$  (Definition 3.1) is an  $\text{type}$ -sampler if for every PPT adversary  $A$  we have*

$$\text{If } \text{type} = \text{odiO}: \mathbb{P} \left[ C_0(x) \neq C_1(x) \mid x \leftarrow S^{C_0(\cdot), C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha) \right] \leq \text{negl}(\lambda),$$

$$\text{If } \text{type} = \text{oiO}: \left| \mathbb{P} \left[ A^{C_0(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha) = 1 \right] - \mathbb{P} \left[ A^{C_1(\cdot)}(1^\lambda, 1^{|C_1|}, \alpha) = 1 \right] \right| \leq \text{negl}(\lambda),$$

where  $(C_0, C_1, \alpha) \leftarrow S(1^\lambda)$ .<sup>10</sup>

**Definition 4.2 (Oracle-differing-input and oracle-indistinguishability obfuscation).** *For  $\text{type} \in \{\text{odiO}, \text{oiO}\}$ , let  $\mathcal{S}$  be an ensemble of  $\text{type}$ -samplers (Definition 4.1). For every  $S \in \mathcal{S}$ , let  $\mathcal{C}^S = \{C_\lambda^S\}_{\lambda \in \mathbb{N}}$  be the ensemble of circuits output by  $S$ . A PPT algorithm  $\text{Obf}$  is a  $(\mathcal{S})$ - $\text{type}$ -obfuscator for the ensemble  $\mathcal{S}$  if the following conditions are satisfied:*

**Correctness.**  $\forall S \in \mathcal{S}, \forall \lambda \in \mathbb{N}, \forall C \in \mathcal{C}_\lambda^S, \forall x \in \{0, 1\}^{\ell_{in}}$ , we have  $C'(x) = C(x)$  where  $C' \leftarrow S \text{Obf}(1^\lambda, C)$ .

**Polynomial slowdown.** *There exists a polynomial  $p$  such that  $\forall S \in \mathcal{S}, \forall C \in \mathcal{C}_\lambda^S$ , we have  $|\text{Obf}(1^\lambda, C)| \leq p(|C|)$ .*

<sup>10</sup> Recall that  $|C_0| = |C_1|$  by definition of sampler (Definition 3.1).

**Indistinguishability.** For every  $S \in \mathcal{S}$ , every PPT adversary  $D$ , we have that

$$|\mathbb{P}[D(1^\lambda, \text{Obf}(1^\lambda, C_0), \alpha) = 1] - \mathbb{P}[D(1^\lambda, \text{Obf}(1^\lambda, C_1), \alpha) = 1]| \leq \text{negl}(\lambda),$$

where  $(C_0, C_1, \alpha) \leftarrow^* \mathcal{S}(1^\lambda)$ .

**Comparing diO-, odiO-, oiO-, and VBB-obfuscation.** We now study the relations between diO, odiO, oiO, and VBB. In order to provide a meaningful comparison, we work in terms of *best-possible universal* obfuscators, i.e., we compare the classes of circuits/samplers that each flavor of obfuscation is able to handle. We start by defining the notion of *best-possible universal type-obfuscator* Obf (for  $\text{type} \in \{\text{diO}, \text{odiO}, \text{oiO}, \text{VBB}\}$ ) whose definition is tied with the (universal) set  $\mathcal{S}_{\text{type}}$  composed of all the type-samplers that can be securely type-obfuscated (as defined in Definitions 4.2 to 3.4).

**Definition 4.3 (Best-possible universal type-obfuscator).** Let  $\text{type} \in \{\text{diO}, \text{odiO}, \text{oiO}, \text{VBB}\}$ . Consider the ensemble  $\mathcal{S}_{\text{type}}$  composed of every type-sampler  $S$  (Definitions 4.1, 3.2 and 3.4) that can be securely type-obfuscated (Definitions 4.2, 3.3 and 3.5), i.e.,

$$\mathcal{S}_{\text{type}} = \{\text{type-sampler } S \mid \exists \text{ Obf s.t. Obf is a } (\{S\})\text{-type-obfuscator}\}.$$

A PPT algorithm Obf is a *best-possible universal type-obfuscator* if Obf is a  $(\mathcal{S}_{\text{type}})$ -type-obfuscator (Definitions 3.3, 4.2 and 3.5).

*Remark 4.4.* There are two technical reasons behind the need of considering only best-possible universal obfuscators, while comparing diO, odiO, oiO, and VBB. First, for any notion of type-obfuscation, it is possible to find two contrived type-obfuscators  $\text{Obf}_0$  and  $\text{Obf}_1$  that result to be incomparable, even within the same flavor of obfuscation. As an example, we could have that  $\text{Obf}_0$  (resp.  $\text{Obf}_1$ ) is able to type-obfuscate  $S_0$  (resp.  $S_1$ ) but not  $S_1$  (resp.  $S_0$ ) where  $S_0, S_1$  are two type-samplers.<sup>11</sup> The same argument holds between different notions. For example, if we consider diO and odiO, we could have that  $\text{Obf}_0$  diO-obfuscates a diO-sampler  $S$  (that in turn, as we will see, is also a odiO-obfuscator) but  $\text{Obf}_1$  does not odiO-obfuscate  $S$ . Also, we can have the symmetric case: there exist two obfuscators  $\text{Obf}'_0$  and  $\text{Obf}'_1$  such that  $\text{Obf}'_1$  odiO-obfuscates  $S$  but  $\text{Obf}'_0$  does not diO-obfuscate  $S$ . Hence by changing the obfuscator we could reach any conclusions: (i) odiO and diO are incomparable, (ii) odiO implies diO, or (iii) diO implies odiO. This clearly does not allow for a meaningful comparison. Definition 4.3 naturally solves the above problem since a best-possible universal type-obfuscator uniquely represents the power of a particular notion of obfuscation, i.e., the set  $\mathcal{S}_{\text{type}}$  of samplers that can be securely type-obfuscated. This allows us to have a meaningful (and unique) formal comparison between diO, odiO, oiO, and VBB.

<sup>11</sup> For instance, we can have that  $S_b$  only outputs circuits whose description starts with a bit  $b$ , and that  $\text{Obf}_b$  rejects any circuit whose description starts with the bit  $1 - b$ .

Second, Definition 4.3 allows us to exclude from the comparison the known impossibility results of VBB [7, 13] (and  $\text{odiO}$ ,  $\text{oiO}$  as we will show in Sect. 6). This is because, instead of quantifying over any possible type-sampler, best-possible universal type-obfuscation is defined over any possible type-sampler that can be type-obfuscated.

In the setting of best-possible universal obfuscation,  $\text{odiO}$  (resp.  $\text{oiO}$ ) is stronger than  $\text{diO}$  since (i) any  $\text{diO}$ -sampler is also an  $\text{odiO}$ -sampler (resp.  $\text{oiO}$ -sampler) and (ii) both  $\text{diO}$  and  $\text{odiO}$  (resp.  $\text{oiO}$ ) have the same indistinguishability-based security definition. The same argument applies to  $\text{odiO}$  and  $\text{oiO}$ , i.e.,  $\text{oiO}$  is stronger than  $\text{odiO}$ .

**Theorem 4.5** ( $\text{oiO} \Rightarrow \text{odiO} \Rightarrow \text{diO}$ ). *For type  $\in \{\text{diO}, \text{odiO}, \text{oiO}\}$ , we have that  $\mathcal{S}_{\text{diO}} \subseteq \mathcal{S}_{\text{odiO}} \subseteq \mathcal{S}_{\text{oiO}}$  where  $\mathcal{S}_{\text{type}}$  as defined in Definition 4.3.*

The proof of this theorem is deferred to full version.

About (best-possible universal)  $\text{odiO}$ -,  $\text{oiO}$ -, and VBB-obfuscation, we have that VBB is stronger than  $\text{odiO}$  (resp.  $\text{oiO}$ ) for two main reasons:

1. VBB leverages a simulation-based definition: any bit of information that can be leaked from an obfuscated circuit  $C$  can be simulated by only having oracle access to  $C$ . On the other hand,  $\text{odiO}$  (resp.  $\text{oiO}$ ) provides a much weaker security guarantee: the obfuscation of two circuits  $C_0, C_1$  (output by an  $\text{odiO}$ -sampler (resp.  $\text{oiO}$ -sampler)) are computationally indistinguishable. This implies that a  $\text{odiO}$ -obfuscator (resp.  $\text{oiO}$ -obfuscator) could leak significant information about the circuit, as long as the leaked information does not help in distinguishing (except with negligible probability) between the obfuscations of  $C_0$  and  $C_1$ .
2. Both VBB and  $\text{odiO}$  (resp.  $\text{oiO}$ ) incorporate the notion of oracle circuits in their definitions. However, oracles are used to define two different concepts. VBB uses oracle circuits to define the amount of information a VBB-obfuscator may leak. Since oracles leak no information (except their input-output behavior), this implies that a VBB-obfuscator does not leak any information, except with negligible probability.

Conversely,  $\text{odiO}$  and  $\text{oiO}$  leverage the notion of oracle circuits to characterize the class of circuits (or samplers) that an  $\text{odiO}/\text{oiO}$ -obfuscator can handle. The definition of security (i.e., the indistinguishability property of Definition 4.2) is independent from the oracles. Both  $\text{odiO}$  and  $\text{oiO}$  “only” guarantee that the information leaked by the obfuscation of two circuits are the same. This does not imply that the  $\text{odiO}/\text{oiO}$ -obfuscated circuits must “behave” as oracles (as required by VBB (Definition 3.5)).

The relation between VBB,  $\text{oiO}$ , and  $\text{odiO}$  is formalized by the following theorem, whose proof is deferred to full version.

**Theorem 4.6** ( $\text{VBB} \Rightarrow \text{oiO}$  and  $\text{VBB} \Rightarrow \text{odiO}$ ). *Let  $S$  be a sampler (Definition 3.1). For  $b \in \{0, 1\}$ , let  $S_b$  be a sampler such that  $(C_b, \alpha) = S_b(1^\lambda; r)$  where  $r \in \{0, 1\}^*$ , and  $(C_0, C_1, \alpha) = S(1^\lambda; r)$ . If  $S_0, S_1 \in \mathcal{S}_{\text{VBB}}$  then  $S \in \mathcal{S}_{\text{type}}$  where  $\mathcal{S}_{\text{VBB}}$  and  $\mathcal{S}_{\text{type}}$  are defined in Definition 4.3.*

By leveraging a similar argument to that used to prove Theorem 4.5, we can demonstrate that any negative result for diO extends to odiO. This because any diO-sampler  $S$  is also an odiO-sampler and, since diO and odiO leverage the same indistinguishability-based definition, if  $S \notin \mathcal{S}_{\text{diO}}$  then  $S \notin \mathcal{S}_{\text{odiO}}$ .<sup>12</sup> The same applies between odiO and oiO, and between oiO and VBB (with respect to samplers as defined in Theorem 4.6).

**Corollary 4.7.** *For type  $\in \{\text{diO}, \text{odiO}, \text{oiO}, \text{VBB}\}$ , let  $\mathcal{S}_{\text{type}}$  be an ensemble of type-samplers as defined in Definition 4.3. The following conditions holds:*

1. *For every diO-sampler  $S$  such that  $S \notin \mathcal{S}_{\text{diO}}$  then  $S \notin \mathcal{S}_{\text{odiO}}$ .*
2. *For every odiO-sampler  $S$  such that  $S \notin \mathcal{S}_{\text{odiO}}$  then  $S \notin \mathcal{S}_{\text{oiO}}$ .*
3. *For every oiO-sampler  $S$  and every pair of VBB-samplers  $(S_0, S_1)$  such that  $(C_b, \alpha) = S_b(1^\lambda; r)$  where  $r \in \{0, 1\}^*$ ,  $(C_0, C_1, \alpha) = S(1^\lambda; r)$  and  $b \in \{0, 1\}$  (as defined in Theorem 4.6), if  $S \notin \mathcal{S}_{\text{oiO}}$  then  $S_0 \notin \mathcal{S}_{\text{VBB}}$  or  $S_1 \notin \mathcal{S}_{\text{VBB}}$ .*

Lastly, odiO (resp. oiO) does not imply VBB, i.e., both odiO and oiO are strictly weaker than VBB. This follows by leveraging two observations. First, Barak et al. [7, Lemma 3.5, Corollary 3.8] have demonstrated that there (unconditionally) exists a distribution of circuits that cannot be VBB-obfuscated (see also Sect. 6.1). This, in turn, implies that there exists a VBB-sampler  $S_0 \notin \mathcal{S}_{\text{VBB}}$ , i.e.,  $S_0$  outputs  $(C, \perp)$  where  $C$  comes from the distribution of [7, Lemma 3.5]. Second, we have that any sampler  $S_1$ , that outputs  $(C_0, C_1, \perp)$  such that  $C_0 = C_1$ , is an odiO-sampler (resp. oiO-sampler) that can be easily odiO-obfuscated (resp. oiO-obfuscated).<sup>13</sup> By combining these two observations, we conclude that if  $S_1$  outputs  $(C_0, C_1, \perp)$  where  $C_0 = C_1$  and  $(C_0, \perp) \leftarrow^* S_0(1^\lambda)$ , it follows that neither  $C_0$  nor  $C_1$  (sampled by  $S_0$ ) can be VBB-obfuscated but  $S_1$  can be odiO-obfuscated (resp. oiO-obfuscated). While this counterexample might be trivial at first sight, it indeed captures the fact that an odiO-/oiO-obfuscator is allowed to reveal any information which is common to the two circuits, as long as this information does not allow to win the respective distinguishing game between the oracles.

**Theorem 4.8 (odiO  $\not\equiv$  VBB and oiO  $\not\equiv$  VBB).** *Let  $S_0$  be a VBB-sampler (Definition 3.4). Consider the odiO-sampler (resp. oiO-sampler)  $S_1$  defined as  $(C_0, C_1, \alpha) = S_1(1^\lambda; r)$  where  $C_0 = C_1$  and  $(C_0, \alpha) = S_0(1^\lambda; r)$  for  $r \in \{0, 1\}^*$ . For type  $\in \{\text{odiO}, \text{oiO}\}$ , there exists a VBB-sampler  $S_0$  such that  $S_0 \notin \mathcal{S}_{\text{VBB}}$  and  $S_1 \in \mathcal{S}_{\text{type}}$  where  $\mathcal{S}_{\text{VBB}}$  and  $\mathcal{S}_{\text{type}}$  as defined in Definition 4.3.*

## 5 Applications of odiO and oiO

In this section, we show that odiO and oiO are able to compile several symmetric key primitives into their corresponding public key versions and designated verifier non-interactive argument systems into their public verifiable version. These

<sup>12</sup> Otherwise, if  $S \in \mathcal{S}_{\text{odiO}}$ , there exists a  $(\{S\})$ -odiO-obfuscator that in turn is also a  $(\{S\})$ -diO-obfuscator.

<sup>13</sup> Indeed, any PPT obfuscator  $\text{Obf}$  that satisfies correctness and polynomial slowdown is a  $(\{S\})$ -odiO-obfuscator (resp.  $(\{S\})$ -oiO-obfuscator), e.g.,  $\text{Obf}$  is the identity function or  $\text{Obf}$  is an iO-obfuscator.

transformations achieve (and use) different flavors of security whose definitions can be found in the full version of this paper. In more details, we demonstrate the following transformations:

**Function-Preserving PV-NIZK from DV-NIZK:** *odiO* is able to compile any designated verifier non-interactive argument system (that satisfies either selective soundness or straight-line knowledge soundness) into its public verifiable version (Sect. 5.1).

**Function-Preserving Signatures from MACs:** *odiO* is able to compile any  $(q)$ -sEUF-sel-CMA MAC into a  $(q)$ -sEUF-sel-CMA signature scheme (full version).

**Format-Preserving Signatures from MACs:** *odiO* is able to compile EUF MAC into a sel-EUF-CMA digital signature scheme, using puncturable PRF (full version).

**Format-Preserving PKE from IV-based SKE:** *odiO* is able to compile semantically secure IV-based SKE (i.e., SKE whose encryption algorithm has the following syntax  $\text{Enc}(k, m; iv) = (iv, c)$ ) into a sel-IND-CPA PKE, using puncturable PRF (full version).

**Function-Preserving PKE from SKE:** *oiO* is able to compile any semantically and sel-IND-CPRA-key secure SKE into a sel-IND-CPA PKE (Sect. 5.2).

Note that transformations that use the puncturable PRFs are only format-preserving whereas the others are fully function-preserving.

We show the first and the last of our applications in detail in the main body; proofs and the remaining applications are deferred to the full version of this work.

### 5.1 From Designated Verifier to Public Verifiable Non-interactive Argument Systems

**Construction 1.** Let  $\Pi^* = (\text{Setup}^*, \text{Prove}^*, \text{Verify}^*)$  and *Obf* be a DV non-interactive argument system for a relation  $\mathcal{R}$  and an obfuscator, respectively. We compile  $\Pi^*$  into a PV non-interactive argument system  $\Pi = (\text{Setup}, \text{Prove}, \text{Verify})$  for the same relation  $\mathcal{R}$  as follows:

**Setup** $(1^\lambda, \mathcal{R})$ : On input the security parameter  $1^\lambda$  and a relation  $\mathcal{R}$ , the setup algorithm computes  $(\text{crs}^*, \text{vrs}^*) \leftarrow_s \text{Setup}^*(1^\lambda, \mathcal{R})$  and outputs  $\text{crs} = \text{crs}^*$  and  $\text{vrs} = \tilde{C}$  where  $\tilde{C} \leftarrow_s \text{Obf}(1^\lambda, C_{\text{vrs}^*}^{\text{Verify}})$  and  $C_{\text{vrs}^*}^{\text{Verify}}$  is depicted in Fig. 2.

**Prove** $(\text{crs}, x, \omega)$ : On input the common reference string  $\text{crs} = \text{crs}^*$ , a statement  $x$ , and a witness  $\omega$ , the prover algorithm outputs  $\pi \leftarrow_s \text{Prove}^*(\text{crs}^*, x, \omega)$ .

**Verify** $(\text{vrs}, x, \pi)$ : On input the verification key  $\text{vrs} = \tilde{C}$ , a statement  $x$ , and a proof  $\pi$ , the verification algorithm returns  $b = \tilde{C}(x, \pi)$ .

Below we establish the following result.

$\frac{C_{\text{vrs}}^{\text{Verify}}(x, \pi)}{\text{return } b = \text{Verify}^*(\text{vrs}, x, \pi)}$	$\frac{S_x(1^\lambda; r)}{(\text{crs}, \text{vrs}) = \text{Setup}^*(1^\lambda; r)}$
$\frac{C_{\text{vrs}, x^*}^{\text{Verify}}(x, \pi)}{\text{If } x = x^*, \text{ return } 0}$	$\text{Set } C_0 = C_{\text{vrs}}^{\text{Verify}}, C_1 = C_{\text{vrs}, x}^{\text{Verify}}, \alpha = \text{crs}$
$\frac{C_{\text{vrs}, \text{td}, r}^{\text{Verify}}(x, \pi)}{\omega = \text{Ext}_1^*(1^\lambda, \text{td}, x, \pi; r)}$	$\text{return } (C_0, C_1, \alpha)$
$\text{If } \text{Verify}^*(\text{vrs}, x, \pi) = 1 \text{ and } (x, \omega) \in \mathcal{R}, \text{ return } 1$	$\frac{S_{\text{Ext}^*}(1^\lambda; r)}{\text{Let } r = (r_0, r_1)}$
$\text{return } 0$	$(\text{crs}, \text{vrs}, \text{td}) = \text{Ext}_0^*(1^\lambda, \mathcal{R}; r_0)$
$\text{Set } C_0 = C_{\text{vrs}}^{\text{Verify}}, C_1 = C_{\text{vrs}, \text{td}, r_1}^{\text{Verify}}, \alpha = \text{crs}$	$\text{return } (C_0, C_1, \alpha)$

**Fig. 2.** The circuits  $C_{\text{vrs}}^{\text{Verify}}$ ,  $C_{\text{vrs}, x^*}^{\text{Verify}}$ ,  $C_{\text{vrs}, \text{td}, r}^{\text{Verify}}$ , and the samplers  $S_x, S_{\text{Ext}^*}$ .  $C_{\text{vrs}}^{\text{Verify}}$  and  $C_{\text{vrs}, x^*}^{\text{Verify}}$  (resp.  $C_{\text{vrs}}^{\text{Verify}}$  and  $C_{\text{vrs}, \text{td}, r}^{\text{Verify}}$ ) are padded to match the size  $\gamma = \max\{|C_{\text{vrs}}^{\text{Verify}}|, |C_{\text{vrs}, x^*}^{\text{Verify}}|\}$  (resp.  $\gamma = \max\{|C_{\text{vrs}}^{\text{Verify}}|, |C_{\text{vrs}, \text{td}, r}^{\text{Verify}}|\}$ ).

**Theorem 5.1.** *Let  $\Pi^*$  and  $\text{Obf}$  as defined in Construction 1. For every  $x \notin \mathcal{L}$ , consider the sampler  $S_x$  depicted in Fig. 2.*

1. *If  $\Pi^*$  satisfies selective soundness then, for every  $x \notin \mathcal{L}$ ,  $S_x$  is an odiO-sampler (Definition 4.1), and*
2. *if  $\text{Obf}$  is a  $(\{S_x\}_{x \notin \mathcal{L}})$ -odiO-obfuscator (Definition 4.2) then the publicly verifiable non-interactive argument system  $\Pi$  of Construction 1 satisfies selective soundness.*

We extend the above result to the case of straight-line knowledge soundness.

**Theorem 5.2.** *Let  $\Pi^*$  and  $\text{Obf}$  as defined in Construction 1.*

1. *If  $\Pi^*$  satisfies straight-line knowledge soundness then the sampler  $S_{\text{Ext}^*}$  of Fig. 2 is an odiO-sampler (Definition 4.1) where  $\text{Ext}^* = (\text{Ext}_0^*, \text{Ext}_1^*)$  is the PPT extractor of  $\Pi^*$ , and*
2. *if  $\text{Obf}$  is a  $(\{S_{\text{Ext}^*}\})$ -odiO-obfuscator (Definition 4.2) then the publicly verifiable non-interactive argument system  $\Pi$  of Construction 1 satisfies straight-line knowledge soundness.*

*Remark 5.3 (On zero-knowledge).* Observe that Construction 1 preserves zero-knowledge if the underlying designated verifier non-interactive argument system  $\Pi^*$  is zero-knowledge. This is straightforward and follows intuitively because Construction 1 only obfuscates  $\text{vrs}$  (that it is known by a malicious verifier against zero-knowledge) and it does not alter  $\Pi^*$ 's Prove. A proof sketch of the zero-knowledge property would be as follows. The simulator for the publicly verifiable case is the same as the one for the designated verifier case. Now assume there exists an adversary  $A_{\text{pv}}$  distinguishing simulated proofs from honest ones. We could then design adversary  $A_{\text{dv}}$  breaking zero-knowledge of the

$C_k^{\text{Enc}}(m, r)$	$S_m(1^\lambda; r)$
<b>return</b> $c = \text{Enc}^*(k, m; r)$	Let $r = (r_0, r_1, r_2)$ $k_0 = \text{KGen}^*(1^\lambda; r_0), k_1 = \text{KGen}^*(1^\lambda; r_1)$ $c = \text{Enc}^*(k_0, m; r_2)$ Set $C_0 = C_{k_0}^{\text{Enc}}, C_1 = C_{k_1}^{\text{Enc}}, \alpha = c$ <b>return</b> $(C_0, C_1, \alpha)$

**Fig. 3.** The circuit  $C_k^{\text{Enc}}$  and the sampler  $S_m$ .  $C_{k_0}^{\text{Enc}}$  and  $C_{k_1}^{\text{Enc}}$  (output by  $S_m$ ) are padded to match the size  $\gamma = \max\{|C_{k_0}^{\text{Enc}}|, |C_{k_1}^{\text{Enc}}|\}$

original scheme. This adversary can in fact internally run  $A_{\text{pv}}$  passing to it the obfuscation  $\text{Obf}(1^\lambda, C_{\text{vrs}}^{\text{Verify}})$ . It can do that because the designated-verifier zero-knowledge has access to  $\text{vrs}$ .

### 5.2 From Semantically and sel-IND-CPRA-key SKEs to sel-IND-CPA PKEs

**Construction 2.** Let  $\Pi^* = (\text{KGen}^*, \text{Enc}^*, \text{Dec}^*)$  and  $\text{Obf}$  be a SKE with message space  $\mathcal{M}$  and an obfuscator, respectively. We compile  $\Pi^*$  into a PKE scheme  $\Pi = (\text{KGen}, \text{Enc}, \text{Dec})$  with message space  $\mathcal{M}$  as follows:

- $\text{KGen}(1^\lambda)$ : On input the security parameter  $1^\lambda$ , the key generation algorithm computes  $k^* \leftarrow \text{KGen}^*(1^\lambda)$  and outputs  $\text{pk} = \tilde{C}$  and  $\text{sk} = k^*$  where  $\tilde{C} \leftarrow \text{Obf}(1^\lambda, C_{k^*}^{\text{Enc}})$  and  $C_{k^*}^{\text{Enc}}$  is depicted in Fig. 3.
- $\text{Enc}(\text{pk}, m; r)$ : On input the public key  $\text{pk} = \tilde{C}$ , a message  $m \in \mathcal{M}$ , and randomness  $r \in \{0, 1\}^*$ , the encryption algorithm outputs  $c = \tilde{C}(m, r)$ .
- $\text{Dec}(\text{sk}, c)$ : On input the secret key  $\text{sk} = k^*$  and a ciphertext  $c$ , the deterministic decryption algorithm returns  $m = \text{Dec}^*(k^*, c)$ .

Below we establish the following result.

**Theorem 5.4.** Let  $\Pi^*$  and  $\text{Obf}$  as defined in Construction 2. For every  $m \in \mathcal{M}$ , consider the sampler  $S_m$  depicted in Fig. 3.

1. If  $\Pi^*$  is sel-IND-CPRA-key then, for every  $m \in \mathcal{M}$ ,  $S_m$  is an oiO-sampler (Definition 4.1), and
2. If  $\Pi^*$  is semantically secure and  $\text{Obf}$  is a  $(\{S_m\}_{m \in \mathcal{M}})$ -oiO-obfuscator (Definition 4.2) then the PKE scheme  $\Pi$  of Construction 2 is sel-IND-CPA.



$C_{k,a,b}^0(x, r)$	$C_{k,a}^1(i, r)$	$C_k^2(c_1, c_2, \odot, r)$
<b>If</b> $x = a$ , <b>return</b> $b$	Let $a = a_1    \dots    a_\lambda$	$x = \text{Dec}_0(k, c_1) \odot \text{Dec}_0(k, c_2)$
<b>return</b> $\text{Enc}_0(k, 0; r)$	<b>return</b> $\text{Enc}_0(k, a_i; r)$	<b>return</b> $\text{Enc}_0(k, x; r)$
$C_{k,a,b,y,e}^3(d_1, \dots, d_\lambda, r)$		$C_{s,(k,a,b,y,e)}^*(\ell, v, r)$
Let $b = b_1    \dots    b_\lambda$	Let $v = (x, i, c_1, c_2, \odot, d_1, \dots, d_\lambda)$	
<b>For</b> $i \in [\lambda]$ <b>do</b> :	$r' = F_1(s, (\ell, v, r))$	
<b>If</b> $\text{Dec}_0(k, d_i) \neq b_i$ ,	<b>If</b> $\ell = 0$ , <b>return</b> $C_{k,a,b}^0(x, r')$	
<b>return</b> $\text{Enc}_0(k, 0; r)$	<b>If</b> $\ell = 1$ , <b>return</b> $C_{k,a}^1(i, r')$	
<b>return</b> $(k, a, y, e)$	<b>If</b> $\ell = 2$ , <b>return</b> $C_k^2(c_1, c_2, \odot, r')$	
	<b>If</b> $\ell = 3$ , <b>return</b> $C_{k,a,b,y,e}^3(d_1, \dots, d_\lambda, r')$	

**Fig. 4.** The circuit  $C_{s,(k,a,b,y,e)}^*$  where  $(s, k, a, b, y, e) \in \{0, 1\}^{5\lambda+1}$  and  $\odot$  is the binary representation of a  $2 \times 2$  table of an arbitrary binary operator (e.g., AND, OR, NOT).

## 6 Extending the Impossibility Results of Barak et al. [7, 8] to the Setting of odiO and oiO

In Sect. 4, we have demonstrated that both odiO and oiO are weaker than VBB and, despite this, these new notions are enough to implement several of the most important applications of VBB (Sect. 5). At this point, the natural question is how weak odiO and oiO are, compared to VBB. In order to give an answer to this question, we investigate whether the impossibility results for VBB (of Barak et al. [7, 8]) extend to either odiO or oiO (or both). Unfortunately, this turned out to be true: As we show in Sect. 6.1, for  $\text{type} \in \{\text{odiO}, \text{oiO}\}$ , there exist a type-sampler that cannot be type-obfuscated (unconditionally).

In addition, Barak et al. [8, Theorem 4.10] have shown that converting an arbitrary SKE into a PKE (by simply obfuscating the SKE’s encryption algorithm together with a symmetric key) is not possible: Indeed, there exists a contrived SKE  $\Pi$  that cannot be obfuscated (as described above) into a PKE. However, such an impossibility result does not apply to our oiO-based transformation from semantically secure and sel-IND-CPRA-key secure SKEs into sel-IND-CPA PKEs (Sect. 5.2) since the contrived SKE  $\Pi$  of [8] is not sel-IND-CPRA-key. Following the same spirit, we study whether a similar argument applies to our format-preserving (deferred to full version) and function-preserving transformations (Construction 2). In this case, we have a negative answer but only for the oiO-based function-preserving transformation (Construction 2): We demonstrate that there exists a SKE  $\Pi$  that is semantically and sel-IND-CPRA-key secure that cannot be converted into a sel-IND-CPA PKE by simply obfuscating the SKE’s encryption algorithm together with a symmetric key, as done by our oiO-based Construction 2. On the other hand, it remains unclear how we can prove a

similar impossibility result for our `odiO`-based format-preserving transformation from SKEs to PKEs (through puncturable PRFs). See full version of this work for more details.

We stress that both our impossibility results leverage similar techniques to that of Barak et al. [7, 8] that we describe in the next sections.

Also, to meet space constraints, the formal proofs of the theorems that appear in next sections are deferred to full version.

### 6.1 Unobfuscatable `odiO`-samplers (Resp. `oiO`-samplers) Exist Unconditionally

We build an ensemble of circuits  $\mathcal{C} = \{C_{s,(k,a,b,y,e)}^*\}$  (indexed by  $(s, k, a, b, y, e) \in \{0, 1\}^{5\lambda+1}$ ) that (i)  $C_{s,(k,a,b,y,e)}^*$  leaks no information when treated as oracles, and (ii) the obfuscation of any  $C_{s,(k,a,b,y,e)}^* \in \mathcal{C}$  allows to extract the hard-coded values  $(k, a, b, y, e)$ . We anticipate that the value  $e \in \{0, 1\}$  will allow us to prove that a circuit  $C_{s,(k,a,b,y,e)}^*$  cannot be `odiO`-obfuscated (resp. `oiO`-obfuscated) (see Sect. 6.1). On the other hand, the value  $y$  is a key of a PRF that is fundamental to build a contrived semantically and sel-IND-CPRA-key secure SKE that cannot be obfuscated (as described in Construction 2) into a sel-IND-CPA PKE (Sect. 6.2). We build such an ensemble  $\mathcal{C}$  (depicted in Fig. 4) by using a similar technique to that of [7, 8] (for more details, we refer the reader to [7, 8]).

In a nutshell,  $C_{s,(k,a,b,y,e)}^*$  (depicted in Fig. 4) is the composition of four circuits  $(C_{k,a,b}^0, C_{k,a}^1, C_k^2, C_{k,a,b,y,e}^3)$  and it is defined with respect to a SKE scheme  $\Pi_0 = (\text{KGen}_0, \text{Enc}_0, \text{Dec}_0)$  and a PRF  $\Pi_1 = (\text{Gen}_1, F_1)$  (required to generate “fresh” randomnesses). On input  $(\ell, v, r)$  where  $v = (x, i, c_1, c_2, \odot, d_1, \dots, d_\lambda)$ ,  $C_{s,(k,a,b,y,e)}^*$  uses  $\ell$  to select which circuit to execute:

1. If  $\ell = 0$ ,  $C_{k,a,b}^0(x, F_1(s, (\ell, v, r)))$  is executed. This circuit presents a trigger input  $a$ . If  $x = a$ ,  $C_{k,a,b}^0(x, F_1(s, (\ell, v, r)))$  returns  $b$ . Otherwise, it returns  $\text{Enc}_0(k, 0; F_1(s, (\ell, v, r)))$ .
2. If  $\ell = 1$ ,  $C_{k,a}^1(i, F_1(s, (\ell, v, r)))$  is executed. This circuit simply outputs the encryption of the  $i$ -th bit of  $a$ , i.e.,  $\text{Enc}_0(k, a_i; F_1(s, (\ell, v, r)))$ .
3. If  $\ell = 2$ ,  $C_k^2(c_1, c_2, \odot, F_1(s, (\ell, v, r)))$  is executed. This circuit allows an evaluator to perform (gate by gate) computations over encrypted inputs. In more detail, it outputs the encryption of the evaluation of  $w \odot z$  (i.e.,  $\text{Enc}_0(k, w \odot z; F_1(s, (\ell, v, r)))$ ) where  $\odot$  is a binary operator, and  $w$  and  $z$  are the bits encrypted by  $c_1$  and  $c_2$ , respectively.
4. If  $\ell = 3$ ,  $C_{k,a,b,y,e}^3(d_1, \dots, d_\lambda, F_1(s, (\ell, v, r)))$  is executed. This is another circuit that presents a trigger input  $b$ . In more detail, if each  $d_i$  is the encryption of the  $i$ -th of  $b$ , the circuit returns  $(k, a, y, e)$ . Otherwise, it returns  $\text{Enc}_0(k, 0; F_1(s, (\ell, v, r)))$ .

Following [7, 8], if the SKE scheme  $\Pi_0$  is IND-CCA1 and  $\Pi_1$  is a secure PRF, then oracle access to  $C_{s,(k,a,b,y,e)}^*$  is computationally indistinguishable to oracle access to a circuit  $\tilde{C}_k$  that, on every input  $(\ell, v, r)$ , it always outputs a

fresh encryption of 0. This is because an adversary only sees ciphertexts and, as a consequence, it cannot distinguish between  $C_{s,(k,a,b,y,e)}^*$  and  $\tilde{C}_k$  unless it guesses the trigger inputs  $a, b \in \{0, 1\}^\lambda$ . As a consequence, this implies that (i) an adversary cannot leak the hardcoded values  $(k, a, b, y, e)$  and, (ii) the pair of circuits  $(C_{s,(k,a,b,y,0)}^*, C_{s,(k,a,b,y,1)}^*)$  are both oracle-differing-input and oracle-indistinguishable circuits (Definition 4.1).

On the other hand, on input  $\tilde{C} \leftarrow_s \text{Obf}(1^\lambda, C_{s,(k,a,b,y,e)}^*)$ , an adversary can easily extract  $(k, a, b, y, e)$ , i.e., the circuit is partially reversible. This can be done as follows:

- Evaluate  $\tilde{C}(1, \cdot, \cdot)$  to get the encryptions  $(c_1, \dots, c_\lambda)$  of the  $a$ 's bits (see Item 2).
- Use  $(c_1, \dots, c_\lambda)$  to compute  $(d_1, \dots, d_\lambda)$  where  $d_i$  is the encryption of  $b$ 's  $i$ -th bit. Observe that this can be done by leveraging  $\tilde{C}(2, \cdot, \cdot)$  to evaluate (gate by gate)  $\tilde{C}(0, \cdot, \cdot) = C_{k,a,b}^0(\cdot, \cdot)$  on  $a$  (see Item 3), and
- Compute  $(k, a, b, y, e)$  by  $\tilde{C}(3, \cdot, \cdot)$  on  $(d_1, \dots, d_\lambda)$  (see Item 4).

The properties of the ensemble  $\mathcal{C}$  are formalized in Theorem 6.1. We highlight that our technique of generating  $\text{Enc}_0$ 's randomness as  $F_1(s, (\ell, v, r))$  (instead of  $F_1(s, (\ell, v))$  as done by Barak et al. [7, 8]) permits to have multiple randomnesses for a fixed pair  $(\ell, v)$ . This allows us to prove a new property (not achieved by [7, 8]) named *input-indistinguishability* that, in turn, is fundamental to prove the impossibility (Sect. 6.2) of converting semantically and sel-IND-CPRA-key secure SKE into sel-IND-CPA PKE. We stress that the ensemble of circuits built by Barak et al. [7, Lemma 3.5] does not satisfy input-indistinguishability.

**Theorem 6.1.** *Let  $\Pi_0 = (\text{KGen}_0, \text{Enc}_0, \text{Dec}_0)$ ,  $\Pi_1 = (\text{Gen}_1, F_1)$ , and  $C_{s,(k,a,b,y,e)}^*$  be a SKE scheme with key space  $\{0, 1\}^\lambda$ , a PRF scheme with key space  $\{0, 1\}^\lambda$ , and the circuit defined in Fig. 4 with respect to  $\Pi_0$  and  $\Pi_1$ , respectively. Then, the ensemble  $\mathcal{C} = \{C_{s,(k,a,b,y,e)}^*\}_{s,k,a,b,y \in \{0,1\}^\lambda, e \in \{0,1\}}$  satisfies the following properties:*

**Oracle-differing-input:** *If  $\Pi_0$  is IND-CCA1 and  $\Pi_1$  is secure then for every PPT adversary  $D$ , we have*

$$\mathbb{P} \left[ C_{s,(k,a,b,y,0)}^*(\ell, v, r) \neq C_{s,(k,a,b,y,1)}^*(\ell, v, r) \right] \leq \text{negl}(\lambda),$$

where  $(\ell, v, r) \leftarrow_s A_{s,(k,a,b,y,0)}^{C_{s,(k,a,b,y,0)}^*}(\cdot, \cdot, \cdot), C_{s,(k,a,b,y,1)}^{C_{s,(k,a,b,y,1)}^*}(\cdot, \cdot, \cdot)(1^\lambda)$ ,  $k \leftarrow_s \text{KGen}_0(1^\lambda)$ ,  $s \leftarrow_s \text{Gen}_1(1^\lambda)$ ,  $y \leftarrow_s \text{Gen}_1(1^\lambda)$ , and  $(a, b) \leftarrow_s \{0, 1\}^{2\lambda}$ .

**Input-indistinguishability:** *If  $\Pi_0$  is IND-CCA1 and IND-CPA-key, and  $\Pi_1$  is secure, then for every  $\ell, v \in \{0, 1\}^*$ , every PPT adversary  $D$ , we have*

$$\left| \mathbb{P} \left[ D_{s_0,(k_0,a_0,b_0,y_0,0)}^{C_{s_0,(k_0,a_0,b_0,y_0,0)}^*}(\cdot, \cdot, \cdot), C_{s_1,(k_1,a_1,b_1,y_1,1)}^{C_{s_1,(k_1,a_1,b_1,y_1,1)}^*}(\cdot, \cdot, \cdot)(1^\lambda, m_0) = 1 \right] - \mathbb{P} \left[ D_{s_0,(k_0,a_0,b_0,y_0,0)}^{C_{s_0,(k_0,a_0,b_0,y_0,0)}^*}(\cdot, \cdot, \cdot), C_{s_1,(k_1,a_1,b_1,y_1,1)}^{C_{s_1,(k_1,a_1,b_1,y_1,1)}^*}(\cdot, \cdot, \cdot)(1^\lambda, m_1) = 1 \right] \right| \leq \text{negl}(\lambda),$$

where  $(a_0, b_0, a_1, b_1) \leftarrow_s \{0, 1\}^{4\lambda}$ ,  $k_j \leftarrow_s \text{KGen}_0(1^\lambda)$  for  $j \in \{0, 1\}$ ,  $s_j \leftarrow_s \text{Gen}_1(1^\lambda)$  for  $j \in \{0, 1\}$ ,  $y_j \leftarrow_s \text{Gen}_1(1^\lambda)$  for  $j \in \{0, 1\}$ , and  $m_d = C_{s_d,(k_d,a_d,b_d,y_d,d)}^*(\ell, v, r_d)$  for  $r_d \leftarrow_s \{0, 1\}^*$  and  $d \in \{0, 1\}$ .

$C_{r,b}^{\text{owf}}(x)$	$S_{\text{owf}}(1^\lambda; r)$
If $x = r$ , return $b$ return 0	Set $C_0 = C_{r,0}^{\text{owf}}, C_1 = C_{r,1}^{\text{owf}}, \alpha = \perp$ return $(C_0, C_1, \alpha)$

**Fig. 5.** The circuit  $C_{r,b}^{\text{owf}}$  and the sampler  $S_{\text{owf}}$ .

**Partial reversibility:** *There exists a PPT algorithm Ext such that for every  $(s, k, a, b, y, e) \in \{0, 1\}^{5\lambda+1}$  and every circuit  $\tilde{C}$  such that  $\tilde{C}(\ell, v, r) = C_{s,(k,a,b,y,e)}^*(\ell, v, r)$  for all  $\ell, v, r \in \{0, 1\}^*$ ,  $\mathbb{P} \left[ (k, a, b, y, e) \leftarrow_s \text{Ext}(1^\lambda, \tilde{C}) \right] = 1$ .*

Theorem 6.1 implies that there exists an odiO-sampler (resp. oiO-sampler)  $\hat{S}$  that cannot be odiO-obfuscated (resp. oiO-obfuscated), if OWFs exist (indeed, OWF implies both IND-CCA1 and IND-CPA-key security of SKE. See full version for more details).

**Corollary 6.2.** *For  $\text{type} \in \{\text{odiO}, \text{oiO}\}$ , if OWFs exist then there exists a type-sampler  $\hat{S}$  (Definition 4.1) such that  $\hat{S} \notin \mathcal{S}_{\text{type}}$  where  $\mathcal{S}_{\text{type}}$  is defined in Definition 4.3.*

Similarly to VBB, both odiO and oiO imply the existence of OWFs. As a consequence, for  $\text{type} \in \{\text{odiO}, \text{odiO}\}$ , a type-unobfuscatable type-sampler exists unconditionally.

**Theorem 6.3.** *Let Obf and  $S_{\text{owf}}$  be an obfuscator and the sampler as defined in Fig. 5. Let  $p(\cdot)$  and  $\mathcal{F} = \{F_\lambda\}_{\lambda \in \mathbb{N}}$  be a polynomial and an ensemble of functions such that  $F_\lambda$  is defined as  $F_\lambda(b, r_0, r_1) = \text{Obf}(1^\lambda, C_{r_0,b}^{\text{owf}}; r_1)$  where  $(b, r_0, r_1) \in \{0, 1\} \times \{0, 1\}^\lambda \times \{0, 1\}^{p(\lambda)}$ . Then, the following statements hold:*

1.  $S_{\text{owf}}$  is an odiO-sampler (resp. oiO-sampler), and
2. if Obf is a  $(\{S_{\text{owf}}\})$ -odiO-obfuscator (resp.  $(\{S_{\text{owf}}\})$ -oiO-obfuscator) then  $F_\lambda \in \mathcal{F}$  is a OWF.

**Corollary 6.4.** *For  $\text{type} \in \{\text{odiO}, \text{oiO}\}$ , there exists (unconditionally) a type-sampler  $S$  such that  $S \notin \mathcal{S}_{\text{type}}$  where  $\mathcal{S}_{\text{type}}$  as defined in Definition 4.3.*

The above corollary follows by combining Corollary 6.2 and Theorem 6.3, i.e., either  $S_{\text{owf}} \notin \mathcal{S}_{\text{type}}$  or  $\hat{S} \notin \mathcal{S}_{\text{type}}$  (for  $\text{type} \in \{\text{odiO}, \text{odiO}\}$ ) where  $S_{\text{owf}}$  and  $\hat{S}$  defined in Fig. 5 and Corollary 6.2, respectively.

## 6.2 Impossibility of Obfuscating Semantically and sel-IND-CPRA-key Secure SKE into sel-IND-CPA Secure PKE Schemes

We now demonstrate that it is inherently impossible to convert a semantically secure and sel-IND-CPRA-key SKEs into sel-IND-CPA PKEs by simply obfuscating the SKE’s encryption algorithm, as described in our oiO-based Construction 2. We prove this by leveraging a similar technique to that of [8]: We construct a SKE  $\Pi^*$  that satisfies semantic and sel-IND-CPRA-key security that,

when obfuscated into a PKE (as described in Sect. 5.2), the latter results to be completely insecure. By leveraging the ensemble  $\mathcal{C}$  of Theorem 6.1, a PRF  $\overline{\Pi} = (\overline{\text{Gen}}, \overline{\text{F}})$ , and a semantically and sel-IND-CPRA-key secure SKE scheme  $\widetilde{\Pi} = (\widetilde{\text{KGen}}, \widetilde{\text{Enc}}, \widetilde{\text{Dec}})$ , we build the contrived SKE  $\Pi^*$  as follows:

$$\text{Enc}^*(k^*, (\ell, v); r) = (\widetilde{\text{Enc}}(\widetilde{k}, (\ell, v); r), C_{s,(\widehat{k},a,b,y,e)}^*(\ell, v, r), \overline{\text{F}}(y, (\ell, v, r)) \oplus \widetilde{k}) \quad (1)$$

where  $k^* = (\widehat{k}, \widetilde{k}, s, a, b, y, e)$ .

$\Pi^*$  is a semantically and sel-IND-CPRA-key secure SKE for the following reasons. First, as described in Sect. 6.1, oracle access to the circuit  $C_{s,(\widehat{k},a,b,y,e)}^* \in \mathcal{C}$  is computationally indistinguishable from having oracle access to a circuit  $\widetilde{C}_k$  that always returns encryptions of 0. Hence, this implies that  $C_{s,(\widehat{k},a,b,y,e)}^*$  does not leak the message  $(\ell, v)$  and that an adversary cannot leak any information about  $(\widehat{k}, a, b, y, e)$ . Second, conditioned to the above observation, the semantic security of  $\Pi^*$  easily follows from the semantic security of  $\widetilde{\Pi}$  and the security of  $\overline{\Pi}$ . Third, as for the sel-IND-CPRA-key security of  $\Pi^*$ , it follows from sel-IND-CPRA-key security of  $\widetilde{\Pi}$ , the security of  $\overline{\Pi}$ , and the fact that  $\mathcal{C}$  satisfies input-indistinguishability (see Theorem 6.1).

On the other hand, when  $\text{Enc}^*$  is obfuscated (as in Construction 2), an adversary can exploit the partial reversibility of  $\mathcal{C}$  (Theorem 6.1) to extract  $y$  and, in turn, the key  $\widetilde{k}$  that is used to encrypt the message  $m = (\ell, v)$ . Below, we report the formal result.

**Theorem 6.5.** *If OWFs exist then the following statements hold:*

1. *there exist a SKE  $\Pi^*$  such that  $\Pi^*$  is semantically secure, sel-IND-CPRA-key, and*
2. *the PKE scheme  $\Pi = (\text{KGen}, \text{Enc}, \text{Dec})$  (output by applying to  $\Pi^*$  the transformation defined in Construction 2) is not sel-IND-CPA (Theorem 5.4).*

We stress that the above result improves the impossibility result of Barak et al. [8] since ours apply to the smaller class of SKEs (i.e., SKEs with stronger notions of security) that satisfy sel-IND-CPRA-key security.

Also, all our odiO-based transformations remain plausible as the impossibility result do not seem to extend. We provide more details in the full version of this work.

**Acknowledgments.** The authors would like to thank the anonymous reviewers for useful feedback. The research described in this paper received funding from: the Concordium Blockchain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions’s Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

## References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptol. ePrint Arch. **2013**, 689 (2013)

2. 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. LNCS, vol. 11694, pp. 284–332. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26954-8\\_10](https://doi.org/10.1007/978-3-030-26954-8_10)
3. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-47989-6\\_15](https://doi.org/10.1007/978-3-662-47989-6_15)
4. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive (2015)
5. Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 26–51. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54242-8\\_2](https://doi.org/10.1007/978-3-642-54242-8_2)
6. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-55220-5\\_13](https://doi.org/10.1007/978-3-642-55220-5_13)
7. 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](https://doi.org/10.1007/3-540-44647-8_1)
8. Barak, B., et al.: On the (im) possibility of obfuscating programs. J. ACM (JACM) **59**(2), 1–48 (2012)
9. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40084-1\\_23](https://doi.org/10.1007/978-3-642-40084-1_23)
10. Bellare, M., Stepanovs, I., Tessaro, S.: Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 102–121. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-45608-8\\_6](https://doi.org/10.1007/978-3-662-45608-8_6)
11. Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 792–821. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49896-5\\_28](https://doi.org/10.1007/978-3-662-49896-5_28)
12. Bitansky, N., Canetti, R.: On strong simulation and composable point obfuscation. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 520–537. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14623-7\\_28](https://doi.org/10.1007/978-3-642-14623-7_28)
13. Bitansky, N., et al.: The impossibility of obfuscation with auxiliary input or a universal simulator. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 71–89. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44381-1\\_5](https://doi.org/10.1007/978-3-662-44381-1_5)
14. Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O.: On virtual grey box obfuscation for general circuits. Algorithmica **79**(4), 1014–1051 (2017)
15. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), pp. 171–190. IEEE Computer Society (2015)
16. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. J. ACM (JACM) **65**(6), 1–37 (2018)
17. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54242-8\\_3](https://doi.org/10.1007/978-3-642-54242-8_3)

18. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 236–261. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48800-3\\_10](https://doi.org/10.1007/978-3-662-48800-3_10)
19. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Candidate iO from homomorphic encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 79–109. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_4](https://doi.org/10.1007/978-3-030-45721-1_4)
20. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54242-8\\_1](https://doi.org/10.1007/978-3-642-54242-8_1)
21. Campanelli, M., Francati, D., Orlandi, C.: Structure-preserving compilers from new notions of obfuscations. Cryptology ePrint Archive, Paper 2022/732 (2022). <https://eprint.iacr.org/2022/732>
22. Canetti, R., Kalai, Y.T., Paneth, O.: On obfuscation with random oracles. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 456–467. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46497-7\\_18](https://doi.org/10.1007/978-3-662-46497-7_18)
23. Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 72–89. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-11799-2\\_5](https://doi.org/10.1007/978-3-642-11799-2_5)
24. 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](https://doi.org/10.1007/11818175_5)
25. Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)
26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 40–49. IEEE Computer Society (2013)
27. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica* **79**(4), 1353–1373 (2017)
28. Garg, S., Mahmoody, M., Mohammed, A.: When does functional encryption imply obfuscation? In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 82–115. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70500-2\\_4](https://doi.org/10.1007/978-3-319-70500-2_4)
29. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 736–749 (2021)
30. Goldwasser, S., Kalai, Y.T.: On the impossibility of obfuscation with auxiliary input. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS2005), pp. 553–562. IEEE (2005)
31. Goldwasser, S., Kalai, Y.T.: A note on the impossibility of obfuscation with auxiliary input. *IACR Cryptol. ePrint Arch.* **2013**, 665 (2013)
32. Goldwasser, S., Rothblum, G.N.: On best-possible obfuscation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 194–213. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-70936-7\\_11](https://doi.org/10.1007/978-3-540-70936-7_11)
33. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 612–621. IEEE (2017)

34. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 668–697. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46497-7\\_26](https://doi.org/10.1007/978-3-662-46497-7_26)
35. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 60–73 (2021)
36. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 447–462. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49387-8\\_17](https://doi.org/10.1007/978-3-662-49387-8_17)
37. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24676-3\\_2](https://doi.org/10.1007/978-3-540-24676-3_2)
38. Mahmoody, M., Mohammed, A., Nematihaji, S.: On the impossibility of virtual black-box obfuscation in idealized models. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 18–48. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49096-9\\_2](https://doi.org/10.1007/978-3-662-49096-9_2)
39. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_28](https://doi.org/10.1007/978-3-662-44371-2_28)
40. Pass, R., Shelat, A.: Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 3–17. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49096-9\\_1](https://doi.org/10.1007/978-3-662-49096-9_1)
41. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, pp. 475–484 (2014)
42. Shacham, H., Waters, B.: Compact proofs of retrievability. *J. Cryptol.* **26**(3), 442–483 (2013)
43. Wee, H.: On obfuscating point functions. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, pp. 523–532 (2005)
44. Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 127–156. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-77883-5\\_5](https://doi.org/10.1007/978-3-030-77883-5_5)
45. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 600–611. IEEE (2017)



# Author Index

## A

Alpár, Greg I-652  
Ambrona, Miguel II-306  
Arun, Arasu II-542  
Ateniese, Giuseppe II-63  
Attema, Thomas II-3  
Avitabile, Gennaro I-281

## B

Badrinarayanan, Saikrishna I-376  
Baum, Carsten I-439  
Bellare, Mihir I-223  
Benz, Laurin I-744  
Beskorovajnov, Wasilij I-744  
Bhadauria, Rishabh II-127  
Botta, Vincenzo I-281  
Bouscатиé, Élie I-774  
Brunetta, Carlo II-336  
Bui, Dung II-190

## C

Campanelli, Matteo II-663  
Capitão, Pedro II-3  
Castagnos, Guilhem I-774  
Catalano, Dario I-471  
Chakraborty, Suvaradip II-635  
Chen, Long II-63  
Chvojka, Peter I-500  
Couteau, Geoffroy II-190, II-221

## D

Datta, Pratish I-587  
David, Bernardo I-439  
Davis, Hannah I-223  
Di, Zijing I-223  
Döttling, Nico II-606  
Dowsley, Rafael I-439  
Ducas, Léo I-177  
Ducros, Clément II-221  
Duman, Julien I-65, I-406  
Düzlü, Samed I-95

## E

Eilebrecht, Sarai I-744

## F

Faonio, Antonio II-275  
Feo, Luca De I-345  
Fiore, Dario I-281, I-471  
Fouotsa, Tako Boris I-345  
Fouque, Pierre-Alain II-461  
Francati, Danilo II-63, II-663

## G

Gajland, Phillip II-606  
Ganesh, Chaya II-542  
Garg, Sanjam II-159  
Gay, Romain II-306  
Ge, Jiangxia I-36  
Georgescu, Adela II-461  
Ghosal, Riddhi II-575  
Ghosh, Satrajit II-251  
Gibbons, Shane I-177  
Giunta, Emanuele I-471  
Gu, Dawu I-685, II-429, II-482  
Güneysu, Tim II-94

## H

Hajiabadi, Mohammad II-159  
Han, Shuai I-685, II-482  
Hartmann, Dominik I-406  
Hazay, Carmit II-127  
Héban, Chloé I-312  
Heninger, Nadia I-147  
Heum, Hans II-336  
Hoffmann, Charlotte I-530  
Hoffmann, Clément I-114  
Hofheinz, Dennis II-275  
Hövelmanns, Kathrin I-65  
Hubáček, Pavel I-530

## J

Jager, Tibor I-500  
Jain, Abhishek II-159

Jiang Galteland, Yao II-399  
 Jin, Zhengzhong II-159

**K**

Kamath, Chethan I-530  
 Kiltz, Eike I-65, II-406  
 Kishore, Ravi I-439  
 Kohl, Lisa II-3  
 Kolonelos, Dimitris II-512  
 Krämer, Juliane I-95  
 Krausz, Markus II-94  
 Kunzweiler, Sabrina I-406  
 Kutas, Péter I-345

**L**

Land, Georg II-94  
 Lee, Kang Hoon II-33  
 Lehmann, Jonas I-406  
 Leroux, Antonin I-345  
 Libert, Benoît I-114  
 Liu, Shengli I-685, II-482  
 Liu, Xiangyu I-685, II-482  
 Lokam, Satya II-542  
 Loss, Julian I-554  
 Lysyanskaya, Anna I-251  
 Lyubashevsky, Vadim I-65

**M**

Malavolta, Giulio I-554, II-606  
 Maller, Mary II-512  
 Maram, Varun I-3  
 Masny, Daniel I-376  
 Merz, Simon-Philipp I-345  
 Miao, Peihan II-368  
 Momin, Charles I-114  
 Mopuri, Tushar II-542  
 Mukherjee, Pratyay I-376  
 Müller-Quade, Jörn I-744

**N**

Nayak, Kartik I-554  
 Nielsen, Jesper Buus I-439

**O**

Oechsner, Sabine I-439  
 Orlandi, Claudio II-663  
 Ottenhues, Astrid I-744

**P**

Pal, Tapas I-587  
 Pan, Jiaxin II-399  
 Pandey, Omkant II-159  
 Panny, Lorenz I-345  
 Papadopoulos, Dimitrios II-63  
 Papamanthou, Charalampos I-554  
 Patranabis, Sikhar I-376, II-368  
 Peters, Thomas I-114  
 Pietrzak, Krzysztof I-530  
 Pointcheval, David I-312  
 Pöppelmann, Thomas I-95  
 Prabhakaran, Manoj II-635  
 Prest, Thomas I-205

**Q**

Qian, Chen II-461

**R**

Raghuraman, Srinivasan I-376  
 Richter-Brockmann, Jan II-94  
 Riepel, Doreen I-406  
 Roux-Langlois, Adeline II-461  
 Roy, Lawrence I-714  
 Russo, Luigi II-275  
 Ryan, Keegan I-147

**S**

Sahai, Amit II-575  
 Sanders, Olivier I-774  
 Sarkar, Pratik I-376  
 Schädlich, Robert I-312  
 Schwerdt, Rebecca I-744  
 Seiler, Gregor I-65  
 Shan, Tianshu I-36  
 Shi, Elaine I-622  
 Shiehian, Sina II-159  
 Simkin, Mark II-251  
 Sridhar, Sriram II-542  
 Srinivasan, Shravan I-554  
 Stam, Martijn II-336  
 Standaert, François-Xavier I-114  
 Struck, Patrick I-95  
 Sun, Shi-Feng II-429

**T**

Tang, Qiang II-63  
 Thyagarajan, Sri AravindaKrishnan I-554

**U**

Unruh, Dominique [I-65](#)

**V**

Vanjani, Nikhil [I-622](#)

Venema, Marloes [I-652](#)

Venkitasubramaniam, Muthuramakrishnan  
[II-127](#)

Volkhov, Mikhail [II-512](#)

**W**

Wang, Geng [II-429](#)

Wang, Zhedong [II-429](#)

Waters, Brent [II-575](#)

Watson, Gaven [II-368](#)

Wen, Weiqiang [II-461](#)

Wesolowski, Benjamin [I-345](#)

Wichs, Daniel [II-635](#)

Wu, Wenxuan [II-127](#)

**X**

Xagawa, Keita [I-3](#)

Xu, Jiayu [I-714](#)

Xue, Rui [I-36](#)

**Y**

Yoon, Ji Won [II-33](#)

**Z**

Zhang, Yupeng [II-127](#)